QUBES

OS

ANONIMATO E SEGURANÇA

ISOLAMENTO · COMPARTIMENTAÇÃO · PRIVACIDADE

Wellington Pinto de Oliveira

Wellington P. Oliveira

Qubes OS para Hackers

Anonimato e Segurança

1ª edição
livreebooks.com.br
LivreEbooks

https://www.livreebooks.com.br

OLIVEIRA, Wellington P.
Qubes OS para Hackers : Anonimato e Segurança / Wellington P. Oliveira
livreebooks.com.br : LivreEbooks,.

1. Boas-vindas e como usar este livro. 2. Modelo de ameaças. 3. Por que sistemas tradicionais falham. 4. O que é o Qubes OS. 5. O hypervisor Xen por dentro. 6. Os cinco mecanismos de isolamento. 7. Hardware, download e verificação da mídia. 8. Instalação passo a passo.

Ficha catalográfica

A Joanna Rutkowska e a toda a equipe que concebeu e mantém o Qubes OS — por provarem que segurança por isolamento não é teoria, e sim um sistema real, livre, que protege quem mais precisa.

Agradecimentos

A Deus, antes de tudo, pela vida e pela oportunidade de aprender e de compartilhar o que se aprende.

À comunidade do Qubes OS, que mantém viva — de forma aberta e gratuita — uma das ferramentas mais importantes para a privacidade das pessoas.

A todos que defendem o conhecimento livre, e a cada leitor que escolhe proteger a si mesmo e a quem depende de si. Este livro é para você.

“Segurança não é um produto, mas um processo.”

— Bruce Schneier

Apresentação

Privacidade e segurança digital não deveriam ser privilégio de quem domina a técnica. O Qubes OS — sistema operacional que protege seus usuários isolando cada atividade em compartimentos independentes — é hoje uma das ferramentas mais sólidas para quem vive sob risco real: jornalistas e suas fontes, ativistas, pesquisadores e qualquer pessoa que não aceite que um único erro comprometa toda a sua vida digital. Ainda assim, falta material em português à altura da sua importância.

Este livro nasceu para preencher essa lacuna. Mais do que ensinar a clicar nos botões certos, ele procura provar por que o Qubes entrega um isolamento que sistemas comuns não alcançam — partindo do modelo de ameaças e descendo até o hipervisor, sempre reconhecendo os limites de cada defesa.

Distribuído gratuitamente, este material aspira a se tornar uma referência lusófona sobre o tema. A orientação prática — a quem ele se destina, como está organizado e que convenções adota — abre o primeiro módulo, e é por lá que a leitura começa.

Prefácio

Vivemos um tempo em que cada gesto digital deixa rastro, e em que um único deslize pode expor anos de trabalho, contatos e confidências. Para a maioria, isso é um incômodo; para quem protege fontes, denuncia abusos ou simplesmente recusa a vigilância como condição de existência, pode ser uma questão de liberdade — às vezes, de vida.

A resposta deste livro não é prometer um sistema invulnerável, porque ele não existe. A premissa é outra, e mais forte: todo software tem falhas, e um dia alguma delas será explorada; o que se pode fazer é projetar o sistema para que, quando isso acontecer, o dano fique contido. É essa mudança de mentalidade — da prevenção impossível para a contenção projetada — que o Qubes OS encarna e que as páginas a seguir procuram tornar clara.

Que este texto sirva a quem precisa de proteção real, e a quem se dispõe a protegê-la.

Sumário

Curso Avançado de Qubes OS — Privacidade e Segurança Pessoal

Livro de apoio para treinamento presencial de 35,5 horas. Material orientado a usuários finais (do iniciante ao avançado) com foco em privacidade: jornalistas, ativistas, pesquisadores e qualquer pessoa que leve a sério a segurança pessoal. Cada conceito técnico é ancorado na documentação oficial do Qubes OS e, sempre que aprofundamos além dela, isso vem sinalizado como complemento.

> *Fonte de referência canônica: documentação oficial em doc.qubes-os.org. Citações entre aspas reproduzem o texto original (em inglês) da fonte.*

Como este livro está organizado

O conteúdo segue sete partes, da motivação à operação avançada, com laboratórios guiados (cerca de metade da carga horária é prática). Os módulos podem ser ministrados na ordem apresentada; um cronograma sugerido de quatro dias acompanha o Módulo 0.

Parte I — Por que compartimentalizar

Parte II — A arquitetura que prova o isolamento

Parte III — Colocando o Qubes para rodar

- **Módulo 6 — Primeiro contato com a interface.** Desktop, menu, Qube Manager; bordas coloridas; widgets de dispositivos, rede e atualização.

Parte IV — Operação no dia a dia

- **Módulo 7 — Conceitos centrais.** TemplateVM, AppVM, a partição /rw e bind-dirs; qubes de sistema; qubes descartáveis.
- **Módulo 8 — Criando e usando qubes.** Criar, clonar e remover; copiar texto e arquivos entre qubes; cópia de e para o dom0.
- **Módulo 9 — Templates: software, atualizações e persistência.** Fedora, Debian e minimal; instalação de software; o que persiste; atualização segura; standalones e HVMs.

Parte V — Rede, anonimato e dispositivos

- **Módulo 10 — Rede e firewall.** Topologia físico, sys-net, sys-firewall e AppVM; endereçamento na faixa 10.137.0.x com máscara /32 ponto a ponto; NAT por salto; firewall aplicado no net qube; laboratório de inspeção de IP, máscara e rota.
- **Módulo 11 — Anonimato com Whonix e Tor.** Gateway sys-whonix e workstation anon-whonix; Tor forçado no gateway; isolamento de fluxos; teste de vazamento.
- **Módulo 12 — Dispositivos: USB, armazenamento, câmera e microfone.** Modelo de anexar e desanexar; sys-usb e BadUSB; dispositivos de bloco; PCI passthrough; microfone e câmera sob demanda.
- **Módulo 13 — Qubes descartáveis em profundidade.** Como funcionam; abertura de anexos não confiáveis; navegação descartável; customização.

Parte VI — Segredos, ameaça física e continuidade

- **Módulo 14 — Chaves e segredos.** O conceito de vault sem rede; Split-GPG; Split-SSH; U2F/CTAP proxy; gerenciador de senhas.
- **Módulo 15 — Ameaça física: criptografia, cold boot e evil maid.** LUKS/dm-crypt para dados em repouso; ataque cold boot a dados em uso; por que suspender é pior que desligar; DMA versus remoção física; Anti Evil Maid e atestação por TPM.
- **Módulo 16 — Backup, restauração e migração.** Backup criptografado; o que incluir; restauração e migração; volume backup/revert.
- **Módulo 17 — Atualização e manutenção do sistema.** Atualização isolada do dom0; templates e Whonix; reinstalação de template; saúde do sistema.

Parte VII — Projeto e fechamento

- **Módulo 18 — Projetando seu esquema de compartimentalização.** Padrões de organização; nomes e cores; estudos de caso; oficina prática.
- **Módulo 19 — Solução de problemas, comunidade e próximos passos.** Troubleshooting comum; onde buscar ajuda; QSBs; como manter-se atualizado.

Apêndices

- **A — Glossário oficial do Qubes.**
- **B — Folha de comandos** (qvm-*, qubesctl, qubesdb-read).
- **C — Mapa de citações da documentação oficial** (rastreabilidade).
- **D — Recursos e leituras complementares.**

Módulo 0 — Boas-vindas e como usar este livro

Este material acompanha um treinamento de aproximadamente 35 horas sobre o Qubes OS, voltado a pessoas que levam a sério a própria privacidade e segurança. Ele foi escrito para ser lido de forma autônoma — como um livro — e também para servir de apoio às aulas presenciais. Você não precisa ser especialista para começar: o curso parte do zero e constrói cada conceito sobre o anterior. Mas também não recua diante do detalhe técnico, porque o objetivo não é apenas mostrar *como* usar o Qubes, e sim *provar* por que ele entrega um isolamento que sistemas comuns não conseguem.

A quem se destina

O público deste curso são usuários finais com foco em privacidade: jornalistas que protegem fontes, ativistas sob vigilância, pesquisadores, profissionais que lidam com informação sensível e qualquer pessoa que não aceite que um único clique errado comprometa toda a sua vida digital. Conhecimento prévio de Linux ajuda, mas não é obrigatório — os comandos são explicados conforme aparecem. Ainda assim, quem está dando os primeiros passos no Linux tende a aproveitar mais o curso depois de ganhar alguma familiaridade com um sistema mais simples, como um Fedora ou um Debian comum: o Qubes é feito de peças desses dois, recompensa quem já conhece a diferença entre eles, e não precisa ser a sua primeira experiência fora do Windows ou do macOS. O que se espera de você é disposição para entender o *porquê* das coisas, e não apenas decorar passos.

Esse público não é hipotético: pessoas reais que dependem de privacidade adotaram o Qubes e relataram boas experiências. O caso mais conhecido é o de **Edward Snowden**, que tornou pública a sua escolha em 29 de setembro de 2016: "If you're serious about security, @QubesOS is the best OS available today. It's what I use, and free. Nobody does VM isolation better" — ou seja, para quem leva segurança a sério, o Qubes seria o melhor sistema disponível, gratuito, e ninguém faria isolamento de VM melhor.

No jornalismo, o exemplo mais direto é o de **Micah Lee** — tecnólogo de segurança que ajudou a proteger a comunicação de Snowden, cofundador da Freedom of the Press Foundation e ex-diretor de segurança da informação do *The Intercept*. Ele usa e recomenda o Qubes ("When I use Qubes I feel like a god") e levou o mesmo princípio de isolamento a ferramentas abertas como o Dangerzone, que sanitiza documentos suspeitos.

A adoção institucional confirma a tendência. A **Freedom of the Press Foundation** construiu o **SecureDrop Workstation** sobre o Qubes OS: ele simula um ambiente isolado (*air-gapped*) para que redações inteiras recebam e abram, com segurança, arquivos enviados por **fontes anônimas** — exatamente o cenário de risco deste curso. Empresas voltadas à privacidade, como a **Mullvad** (VPN) e a **Let's Encrypt** (maior autoridade certificadora do mundo), também relatam usar o Qubes na maioria de suas estações de trabalho.

Para quem precisa de **anonimato**, há ainda guias mantidos por e para comunidades sob vigilância — como o "Qubes OS for Anarchists" — que documentam, passo a passo, o uso do sistema por ativistas que não podem se permitir um único vazamento. Em todos esses casos, a motivação é a mesma que move este curso: conter o dano quando — não *se* — algo der errado.

Como o curso está organizado

O conteúdo se divide em sete partes, da motivação à operação avançada:

- **Parte I — Por que compartimentalizar:** o modelo de ameaças e por que sistemas tradicionais falham.
- **Parte II — A arquitetura que prova o isolamento:** o que é o Qubes, o hypervisor Xen e os cinco mecanismos de isolamento.
- **Parte III — Colocando o Qubes para rodar:** hardware, verificação da mídia, instalação e a interface.
- **Parte IV — Operação no dia a dia:** conceitos centrais, criação e uso de qubes, templates.
- **Parte V — Rede, anonimato e dispositivos:** firewall, Whonix/Tor, dispositivos e qubes descartáveis.
- **Parte VI — Segredos, ameaça física e continuidade:** chaves, criptografia e cold boot, backup, manutenção.
- **Parte VII — Projeto e fechamento:** desenhar seu próprio esquema, solução de problemas e próximos passos.

Cada módulo termina com um **laboratório guiado**. Cerca de metade da carga horária do curso é prática: o Qubes só faz sentido nas mãos, e os labs foram desenhados para você comprovar, no sistema real, o que o texto afirma.

Convenções deste livro

Para ler com fluidez, conheça os elementos recorrentes:

- **Citações da documentação oficial.** As definições e afirmações vêm da documentação oficial do Qubes OS (doc.qubes-os.org) e são apresentadas em português, por citação indireta, fiéis ao sentido da fonte — que você pode conferir pelos endereços nas Referências do capítulo.
- **Referências ao final do capítulo.** Cada capítulo fecha com um bloco "Referências do capítulo", reunindo as páginas-fonte com o endereço completo, para você aprofundar.
- **Caixas de aprofundamento.** Trechos marcados como "Aprofundamento" trazem detalhes técnicos que vão além da documentação de usuário (por exemplo, internals do hypervisor Xen). São opcionais: pule-os se quiser apenas o essencial, leia-os se quiser a engenharia por trás.
- **Reforços de privacidade.** Sempre que um conceito técnico se conecta diretamente a um ganho de privacidade ou anonimato, há um parágrafo de reforço explicando o que aquilo significa, na prática, para quem está sob risco. Eles são o fio condutor do curso.
- **Laboratórios.** Numerados por módulo, com passos a executar. Comandos aparecem em blocos de código; preste atenção a *onde* cada comando roda (no dom0 ou dentro de um qube), porque isso é uma questão de segurança.

Montando seu laboratório de estudo

O ideal é praticar em uma instalação real do Qubes OS, em hardware compatível (veja o Módulo 4 para requisitos). Como o Qubes é um hypervisor bare-metal (Tipo 1, Módulo 3B), ele não roda bem "dentro" de outra máquina virtual — virtualização aninhada é frágil e não reproduz o isolamento por hardware que o curso estuda. Por isso, se possível, use um computador dedicado ou secundário para os laboratórios.

Se você ainda não tem hardware disponível, pode acompanhar todo o curso pela leitura e fazer os passos de planejamento (como o Exercício do Módulo 1 e o projeto do Módulo 18), deixando os labs práticos para quando tiver uma máquina. O conteúdo conceitual e a "prova" de segurança não dependem de você ter o sistema instalado.

Cronograma sugerido

Para um curso presencial de quatro dias com cerca de oito horas cada:

- **Dia 1:** Módulos 0 a 5 — fundamentos e instalação.
- **Dia 2:** Módulos 6 a 9 — interface, conceitos, dia a dia e templates.
- **Dia 3:** Módulos 10 a 14 — rede, Whonix, dispositivos, descartáveis e segredos.
- **Dia 4:** Módulos 15 a 19 — ameaça física, backup, manutenção, projeto final e troubleshooting.

Como autoaprendizado, vá no seu ritmo: o importante é fazer os laboratórios, e não apenas ler. A compreensão vem da prática.

Uma palavra antes de começar

O Qubes OS pede um pouco mais de disciplina do que um sistema comum — você vai pensar em "qual qube" antes de "qual programa", vai confirmar cópias entre compartimentos, vai conferir a cor da janela antes de digitar uma senha. Esses pequenos hábitos são o preço da contenção: eles garantem que, quando algo der errado — e a premissa do curso é que um dia dará —, o estrago fique preso em uma única célula. Se você internalizar isso, terá entendido o Qubes. É hora de começar.

Referências do capítulo

- Edward Snowden — endosso ao Qubes OS (Twitter/X, 29/09/2016). https://x.com/Snowden/status/781493632293605376
- Qubes OS — Endorsements (Snowden, Micah Lee, Kenn White, Bill Budington e organizações). https://www.qubes-os.org/endorsements/
- Freedom of the Press Foundation — SecureDrop Workstation, baseado em Qubes OS. https://securedrop.org/
- SecureDrop Workstation — whitepaper técnico (design sobre Qubes). https://media.securedrop.org/media/documents/SD*Qubes*Workstation_Whitepaper.pdf
- AnarSec — Qubes OS for Anarchists (uso por ativistas e pessoas anônimas). https://www.anarsec.guide/posts/qubes/
- Qubes OS — Introduction. https://doc.qubes-os.org/en/latest/introduction/intro.html
- Qubes OS — Getting started. https://doc.qubes-os.org/en/latest/introduction/getting-started.html
- Qubes OS — FAQ. https://doc.qubes-os.org/en/latest/introduction/faq.html

Módulo 1 — Modelo de ameaças

Antes de instalar qualquer ferramenta de segurança — incluindo o Qubes OS — existe uma pergunta que precisa vir primeiro: *do que, exatamente, você está se protegendo?* Pular essa pergunta é o erro mais comum de quem busca privacidade. A pessoa baixa um navegador "anônimo", liga uma VPN, instala um antivírus e sente-se segura, sem nunca ter definido contra quem ou contra o quê. Segurança sem um modelo de ameaças é superstição: gestos que tranquilizam, mas não necessariamente protegem.

Modelar ameaças é o ato de tornar explícito o que costuma ficar no automático. É responder, de forma honesta e específica, quatro perguntas:

- o que você quer proteger;
- de quem;
- quão graves são as consequências de uma falha;
- quanto esforço está disposto a investir.

Só depois dessas respostas faz sentido escolher ferramentas. O Qubes é uma ferramenta poderosa, mas poder não é o mesmo que adequação: ele resolve bem certos problemas e não resolve outros, e este capítulo é o que vai permitir que você reconheça a diferença.

Vale ver isso num caso concreto. O exemplo a seguir é uma **simulação didática** — uma reconstrução hipotética, e não um relato das decisões reais de Edward Snowden. Ele serve só para mostrar as quatro perguntas aplicadas a uma situação reconhecível: a de alguém que, em 2013, precisou entregar documentos secretos a jornalistas sabendo que enfrentaria o adversário mais capaz que existe.

- **O que proteger?** Antes de tudo, a identidade e a segurança das pessoas envolvidas — as fontes, os jornalistas que receberiam o material, familiares e contatos. Depois, os próprios documentos, a localização física de quem os manuseia e a integridade das comunicações.
- **De quem proteger?** De um adversário estatal com recursos praticamente ilimitados: agências de inteligência capazes de vigiar a rede em escala global, explorar falhas desconhecidas (0-days), interceptar dispositivos e coagir provedores. Não é o ladrão de notebook do dia a dia — é o topo da escala de ameaças.
- **Quão graves são as consequências?** Máximas e irreversíveis: a exposição significaria prisão, risco à vida e à liberdade de fontes e colaboradores, e o fracasso de toda a operação. Não há "tentar de novo".
- **Quanto esforço investir?** Praticamente todo o disponível. Nesse perfil, abre-se mão de quase toda conveniência: máquinas dedicadas, sistemas amnésicos, anonimato de rede via Tor, criptografia forte, verificação de cada ferramenta e separação rígida entre identidades.

Repare o que o exercício revela: as respostas 2 e 4 — um adversário que observa a rede e a disposição de pagar o preço da inconveniência — apontam para **isolamento somado a anonimato**, não apenas um dos dois. É justamente a combinação que o Qubes oferece junto com o Whonix, detalhada nos próximos módulos. Um modelo mais modesto — proteger fotos pessoais de um colega curioso, por exemplo — levaria a escolhas bem mais leves. A ferramenta certa nasce da pergunta certa.

> *Reforço de privacidade. Mesmo num caso extremo, o método é o mesmo que vale para você: tornar explícito o que está em jogo, contra quem, com que gravidade e a que custo. A diferença entre o usuário comum e alguém na situação de Snowden não está no método — está nas respostas. Por isso o Exercício 1, ao final deste módulo, pede o seu modelo: é ele que calibra cada decisão do curso ao seu risco real.*

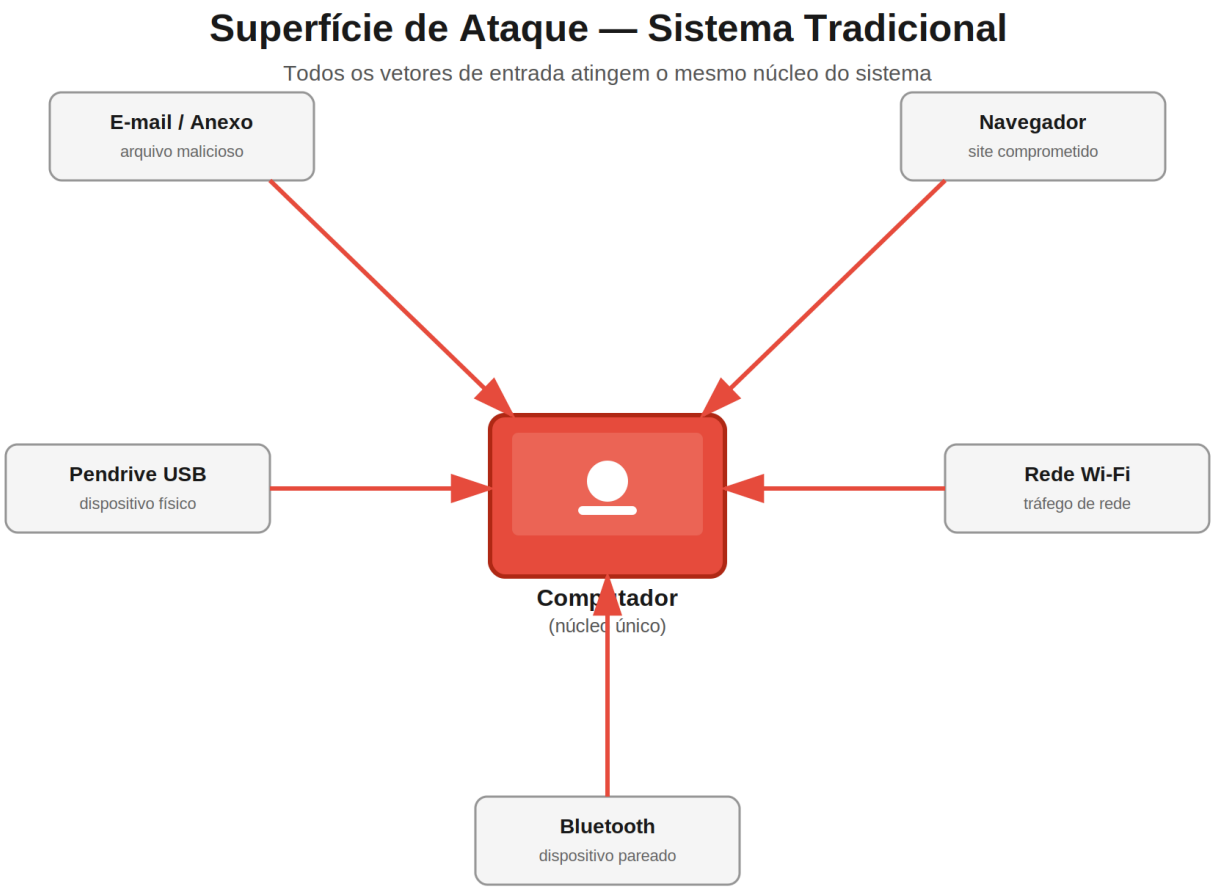
Ameaça, risco e superfície de ataque

Três termos formam o vocabulário básico deste curso, e vale fixá-los com precisão, porque eles serão usados o tempo todo.

Uma **ameaça** é um evento adverso que você quer evitar: o roubo do notebook, a interceptação de uma conversa, a identificação de uma fonte jornalística, a infecção por um malware vindo de um anexo. A ameaça é sempre concreta e ligada a um adversário ou a uma circunstância.

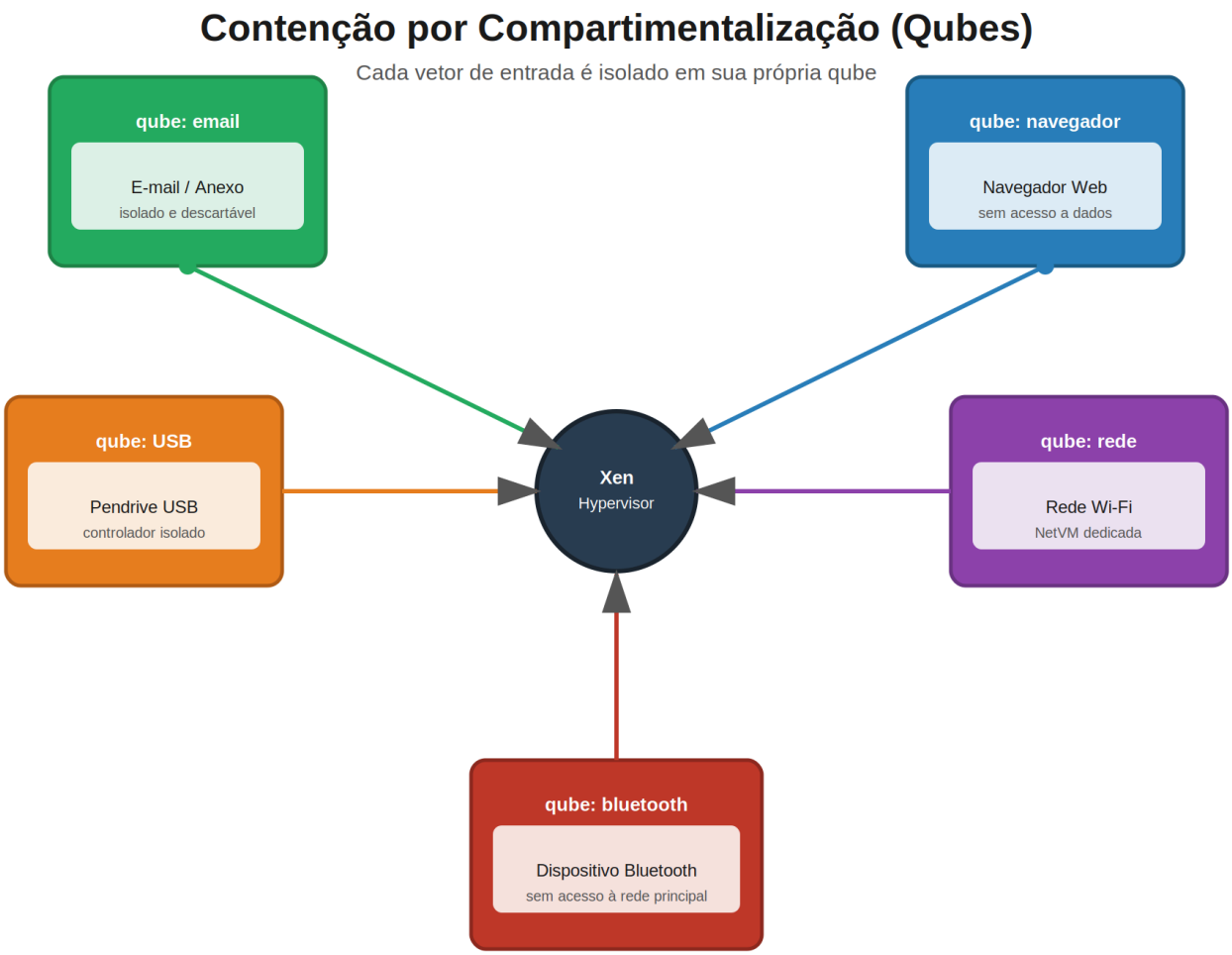
O **risco** é a combinação de duas coisas: a probabilidade de a ameaça se concretizar e a gravidade do dano se ela se concretizar. Uma ameaça muito grave mas improvável pode merecer menos atenção do que uma ameaça moderada e cotidiana. Pensar em risco evita dois extremos igualmente ruins: a paranoia que paralisa e a despreocupação que expõe.

A **superfície de ataque** é a soma de todos os pontos por onde um adversário pode tentar agir: cada programa que processa dados de fora, cada porta de rede aberta, cada dispositivo conectado, cada pendrive que você espeta. A ideia central da segurança moderna — e do Qubes em particular — é que não dá para tornar essa superfície perfeita, mas dá para reduzi-la e compartimentá-la, de modo que uma falha em um ponto não contamine todo o resto.



Uma falha em qualquer vetor pode comprometer todo o sistema

Repare, na imagem acima, como tudo converge para um único sistema operacional: navegador, e-mail, banco, arquivos e dispositivos compartilham o mesmo espaço de confiança. Cada seta — um anexo, um site, um pendrive, a rede Wi-Fi — aponta para o mesmo lugar. Basta que um desses vetores tenha sucesso para que o atacante alcance tudo o que é seu, porque não há parede interna separando uma atividade da outra.



No Qubes, os mesmos vetores chegam primeiro a compartimentos dedicados, e não ao seu sistema: o **sys-usb** recebe os dispositivos USB e o **sys-net** recebe a rede — justamente os dois pontos mais expostos e historicamente mais inseguros. Esses qubes seguram o hardware perigoso longe do resto. Se um pendrive malicioso ou um ataque de rede tiver sucesso, o estrago fica preso naquele compartimento, sem contaminar seus arquivos, suas senhas ou suas fontes. A mesma superfície de ataque continua existindo — mas agora ela é fatiada e contida.

A premissa fundadora: "todo software tem bugs"

O Qubes parte de uma premissa que, uma vez aceita, muda toda a forma de pensar segurança. A documentação a expressa sem rodeios: todo software contém bugs. Não é pessimismo; é estatística. Qualquer programa não trivial tem defeitos, e alguns desses defeitos são exploráveis por um atacante. Quanto maior e mais complexo o software, mais bugs ele esconde.

A maioria das abordagens tradicionais de segurança aposta no contrário: tentam impedir que os bugs sejam explorados, detectar o ataque quando ele acontece, ou remendar a falha depois que ela é descoberta. É uma corrida sem fim, na qual o defensor precisa acertar sempre e o atacante precisa acertar uma única vez.

A história recente confirma a premissa: nenhuma tecnologia, por mais central ou bem mantida que seja, está imune. Cinco casos amplamente divulgados, em projetos e fornecedores de primeira linha:

- **OpenSSH** — a falha "regreSSHion" (CVE-2024-6387, 2024) permitia execução remota de código como root no servidor sshd, com milhões de instâncias expostas na internet.
- **Linux** — a "Dirty Pipe" (CVE-2022-0847, 2022) deixava um usuário sem privilégios sobrescrever arquivos somente leitura e se tornar root pelo kernel.
- **Windows** — uma falha na pilha de rede TCP/IP sobre IPv6 (CVE-2024-38063, 2024) permitia execução remota de código "wormable", sem qualquer interação do usuário.
- **Cisco** — no IOS XE, a CVE-2023-20198 (2023, gravidade máxima 10,0) foi explorada em massa, comprometendo mais de 10 mil dispositivos de rede.
- **Fortinet** — no FortiOS SSL-VPN, a CVE-2024-21762 (2024) permitia execução remota de código sem autenticação e foi explorada ativamente.

O que esses casos têm em comum não é desleixo — é a regra: software complexo falha. A diferença de valor entre uma tecnologia e outra não está em "nunca falhar", e sim na transparência com que cada falha é reconhecida e divulgada (um CVE público, um aviso de segurança) e na abordagem que limita o estrago quando ela ocorre.

O Qubes assume a derrota nessa corrida específica e muda o jogo. Partindo do princípio de que o software *vai* falhar, a pergunta deixa de ser "como impedir a falha?" e passa a ser "como garantir que a falha, quando ocorrer, fique contida?". Essa inversão é o que justifica toda a arquitetura estudada no capítulo sobre o Xen: a TCB pequena, o dom0 sem rede, a separação por hardware. Não são truques para impedir bugs; são estruturas para conter o estrago dos bugs que certamente existem.

Confinar, controlar e conter o dano

Da premissa anterior nasce a estratégia, resumida pela própria documentação em três verbos: confinar, controlar e conter o dano. Essa é a frase que define o Qubes melhor do que qualquer outra, e vale destrinchá-la.

Confinar é dar a cada atividade seu próprio compartimento isolado, de modo que ela não tenha acesso ao que não lhe diz respeito. Seu navegador de notícias não precisa enxergar os arquivos do seu trabalho; seu cliente de e-mail pessoal não precisa acessar suas chaves de criptografia. Cada qube é uma cela.

Controlar é mediar, de forma explícita, qualquer interação entre os compartimentos. No Qubes, copiar um texto ou um arquivo de um qube para outro é uma ação deliberada, que o usuário autoriza; nada vaza de um compartimento para outro por acidente.

Conter o dano é a consequência das duas anteriores: quando — não *se* — um compartimento for comprometido, o estrago para ali. A Introdução do projeto traduz isso em uma imagem que fala diretamente ao público deste livro: você pode fazer tudo no mesmo computador físico sem ter de se preocupar que um único ciberataque bem-sucedido derrube toda a sua vida digital de uma só vez. Um ataque bem-sucedido contra o seu navegador descartável não alcança suas fontes, suas senhas, seus documentos.

> *Reforço de privacidade. Para um jornalista, um ativista ou alguém em situação de risco, "conter o dano" não é abstração técnica: é a diferença entre um navegador comprometido revelar uma aba e revelar a identidade de uma fonte, a localização de um contato ou o conteúdo de uma investigação inteira. A compartimentalização existe justamente para que um deslize — um anexo malicioso, um site armadilhado — não se transforme na exposição de tudo o que você precisa manter privado.*

Quem é você no mapa de risco

Ferramentas de segurança não são boas ou ruins em abstrato; são adequadas ou inadequadas a um perfil de risco. O Qubes foi pensado, segundo a documentação, para servir tanto a pessoas vulneráveis e ativamente visadas quanto a organizações que precisam de garantias fortes de isolamento. Vale situar alguns perfis típicos do público deste livro:

O **jornalista** que protege fontes precisa garantir que a identidade de quem fala com ele não vaze, mesmo que seu computador processe arquivos enviados por desconhecidos. Sua ameaça inclui adversários com recursos: governos, corporações, grupos organizados.

O **ativista** opera muitas vezes sob vigilância e sob risco de apreensão do equipamento. Para ele importam tanto a contenção digital quanto a proteção física do dispositivo — tema retomado no módulo sobre cold boot e criptografia.

O **pesquisador de segurança** ou analista de malware precisa abrir, de propósito, arquivos perigosos, sem contaminar o resto do sistema. O isolamento é, para ele, ferramenta de trabalho diária.

O **usuário pessoal preocupado com privacidade** quer simplesmente que um clique errado em um link não comprometa seu banco, suas fotos e suas conversas de uma só vez.

Vale, por fim, situar três tipos clássicos de hacker, porque a compartimentalização aparece em todos eles:

- **Blackhat** — usa suas habilidades de forma ilícita, atuando no submundo (incluindo a darkweb). Para este livro, é sobretudo o *adversário* cujas técnicas justificam as defesas deste livro — não o público a quem ele se destina.
- **Whitehat** — o profissional de segurança que protege clientes de forma legal. Apoia-se na individualização por VM para isolar cada cliente, engajamento ou alvo de teste, mantendo separados — e seguros — ambientes que jamais podem se misturar. É um dos usos mais naturais do Qubes.
- **Hacktivista** — emprega habilidades técnicas em causas políticas, com frequência enfrentando sistemas governamentais opressores. Seu perfil de risco se aproxima do ativista: precisa de isolamento e, quase sempre, de anonimato forte (Whonix/Tor).

> *Reforço de privacidade. Definir seu perfil é o que determina quanto anonimato você precisa, e não só quanta segurança. Um analista de malware pode não se importar em ser identificado, desde que o malware não escape do compartimento; já uma fonte jornalística pode precisar de anonimato absoluto, o que exige Whonix e Tor, e não apenas isolamento. Segurança e anonimato são objetivos distintos: o Qubes entrega o primeiro por padrão e oferece o segundo quando você o configura explicitamente. Não confunda os dois.*

Exercício 1 — Escrevendo seu modelo de ameaças

Este exercício é individual e deve ser feito por escrito, à mão ou em um qube de anotações. Não há resposta certa; há a *sua* resposta, que guiará suas escolhas pelo resto do curso. Reserve de quinze a vinte minutos.

Responda, com o máximo de especificidade possível:

1. **O que você quer proteger?** Liste seus "ativos": documentos, identidade, contatos, fontes, chaves, contas financeiras, localização, histórico de navegação. Seja concreto

— "minhas conversas com a fonte X", não apenas "minha privacidade".

2. **De quem você quer se proteger?** Liste os adversários plausíveis: um ladrão oportunista, um ex-parceiro, um colega curioso, um concorrente, um provedor de internet, uma autoridade. Para cada um, estime os recursos que ele tem.
3. **Quão graves são as consequências se você falhar?** Para cada ativo, imagine o pior caso realista. A gravidade varia: vazar uma lista de compras é diferente de vazar a identidade de uma fonte.
4. **Quão provável é que você precise se proteger disso?** Separe o cotidiano (anexos, sites, redes públicas) do excepcional (apreensão do equipamento, adversário estatal).
5. **Quanto esforço você está disposto a investir?** Segurança tem custo de conveniência. Reconheça o seu limite com honestidade — um esquema que você não consegue manter é pior do que um esquema modesto que você segue.

Guarde esse documento. Ao final do curso, no módulo sobre como projetar seu esquema de compartimentalização, esse documento é retomado e essas respostas se transformam em um desenho concreto de qubes — quantos, com que finalidade, com que nível de confiança e com ou sem anonimato.

> *Reforço de privacidade. Note que as perguntas 2 e 4 são as que definem sua necessidade de anonimato. Se entre seus adversários há quem possa observar de onde você se conecta ou com quem você fala — e não apenas o conteúdo —, então isolamento não basta: você precisará de anonimato de rede via Tor/Whonix. Tenha essa distinção em mente; ela reaparece em vários módulos.*

Referências do capítulo

- Qubes OS — Introduction (confinar, controlar e conter o dano; um único ciberataque derrubando toda a vida digital). https://doc.qubes-os.org/en/latest/introduction/intro.html
- Electronic Frontier Foundation — Your Security Plan (Surveillance Self-Defense; complementar, fora da doc Qubes). https://ssd.eff.org/en/module/your-security-plan
- OpenSSH "regreSSHion" — CVE-2024-6387 (Qualys Threat Research; complementar). https://blog.qualys.com/vulnerabilities-threat-research/2024/07/01/regresshion-remote-unauthenticated-code-execution-vulnerability-in-openssh-server
- Linux kernel "Dirty Pipe" — CVE-2022-0847 (Max Kellermann; complementar). https://dirtypipe.cm4all.com/
- Windows TCP/IP (IPv6) — CVE-2024-38063 (Microsoft MSRC; complementar). https://msrc.microsoft.com/update-guide/vulnerability/CVE-2024-38063
- Cisco IOS XE Web UI — CVE-2023-20198 (Cisco Talos; complementar). https://blog.talosintelligence.com/active-exploitation-of-cisco-ios-xe-software/
- Fortinet FortiOS SSL-VPN — CVE-2024-21762 (NVD; complementar). https://nvd.nist.gov/vuln/detail/CVE-2024-21762

Módulo 2 — Por que sistemas tradicionais falham

É comum culpar o usuário pelas falhas de segurança: "clicou no link errado", "não atualizou", "usou senha fraca". Há verdade nisso, mas a culpa de fundo é arquitetural. Os sistemas operacionais tradicionais — Windows, macOS, as distribuições Linux comuns — foram projetados sob uma premissa que hoje sabemos frágil: a de que todos os programas que você executa rodam, essencialmente, no mesmo espaço de confiança. Um único erro, em um único programa, pode comprometer tudo.

Este módulo examina por que isso acontece e por que as defesas usuais — antivírus, máquinas virtuais comuns, até computadores separados — não resolvem o problema de raiz. O objetivo não é depreciar essas abordagens, mas mostrar com precisão onde cada uma falha, para que a proposta do Qubes apareça não como mais uma opção, e sim como uma resposta a um problema estrutural.

O kernel monolítico: um bug, comprometimento total

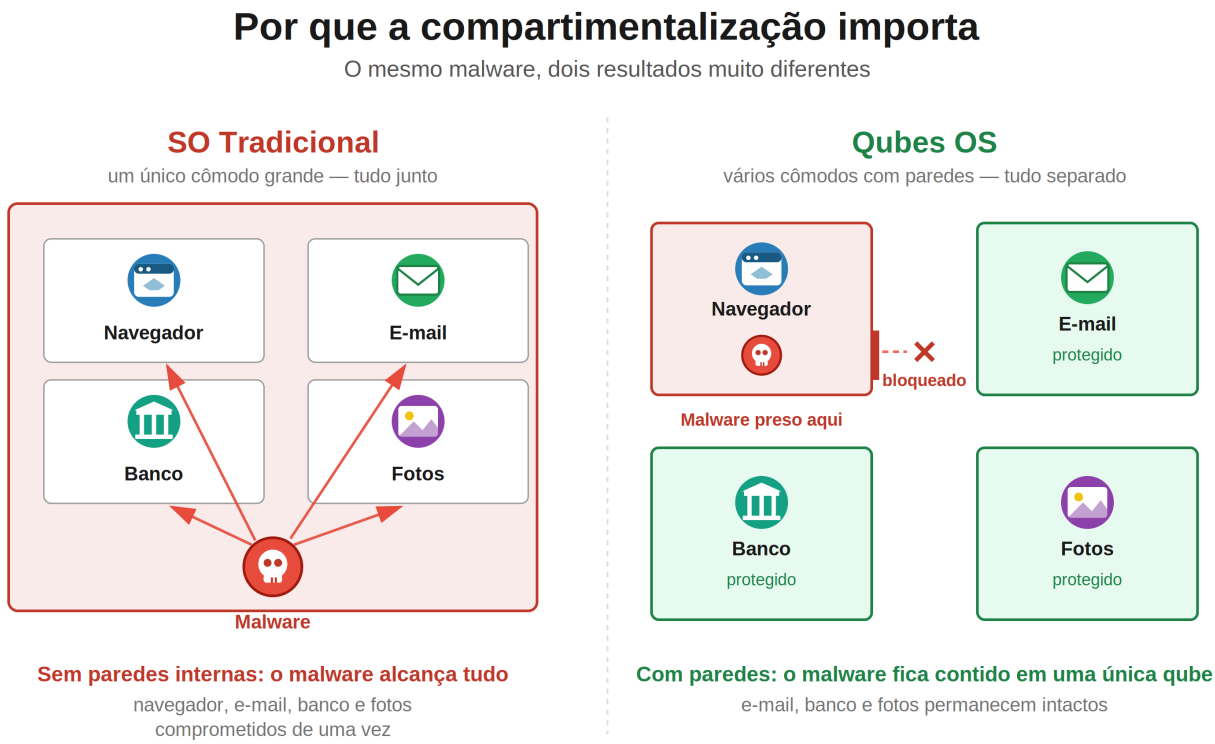
No coração de um sistema operacional tradicional está o kernel monolítico. Ele é gigantesco — dezenas de milhões de linhas de código — e roda com o privilégio máximo da máquina. Dentro dele convivem o gerenciamento de memória, o sistema de arquivos, a pilha de rede e, crucialmente, milhares de drivers para todo tipo de dispositivo.

O problema é que tudo isso compartilha o mesmo espaço de confiança. Um bug em um driver de Wi-Fi, em um decodificador de áudio ou na pilha de Bluetooth não fica restrito àquele componente: como todos rodam no kernel, explorá-lo costuma dar ao atacante o controle do sistema inteiro. Não existe parede interna. A superfície de ataque do kernel é imensa, e qualquer ponto dela que ceda derruba toda a estrutura.

Some-se a isso o modelo de usuário. Em um SO tradicional, todos os seus aplicativos rodam sob a sua conta e, portanto, têm acesso a tudo o que é seu: seu navegador pode ler os arquivos do seu trabalho, seu leitor de PDF pode acessar suas fotos, seu cliente de e-mail pode varrer suas chaves. Eles *não precisam* desse acesso, mas o têm, porque a arquitetura não os separa. Basta um deles ser comprometido.

A literatura clássica de sistemas operacionais ajuda a situar por que o modelo monolítico domina — e onde ele falha. Ao apresentar o "zoológico" de sistemas operacionais (do mainframe ao embarcado, do servidor ao cartão inteligente), Tanenbaum, em *Sistemas Operacionais Modernos*, cataloga as arquiteturas internas possíveis. No **sistema monolítico**, todo o SO roda como um único programa em modo núcleo, e qualquer procedimento pode chamar qualquer outro: é rápido, mas não há fronteira interna, de modo que um defeito em qualquer ponto compromete o todo — é o modelo de Linux, Windows e macOS. Nos **sistemas em camadas**, o SO se organiza em níveis hierárquicos, cada um usando apenas os serviços do nível imediatamente abaixo (a ideia pioneira do sistema THE, de Dijkstra, em 1968). No **micronúcleo (microkernel)**, o núcleo é reduzido ao mínimo — escalonamento, memória básica e troca de mensagens — e os serviços (drivers, sistema de arquivos, pilha de rede) rodam isolados, em modo usuário, conversando por mensagens no modelo cliente-servidor; assim, um driver que falha pode ser reiniciado sem derrubar o sistema, princípio em torno do qual o próprio Tanenbaum construiu o **MINIX 3**, voltado à confiabilidade.

Essa divisão alimentou um debate célebre: em 1992, Tanenbaum declarou o kernel monolítico uma tecnologia ultrapassada, e Linus Torvalds defendeu o Linux monolítico por desempenho e simplicidade. No desktop, o desempenho fez o monolítico vencer — e é daí que esses sistemas herdam o problema descrito acima: tudo no mesmo espaço de confiança, sem paredes internas. O Qubes não tenta vencer esse debate reescrevendo o kernel; aplica a mesma intuição do micronúcleo — separar o que não precisa estar junto — uma camada **abaixo** do sistema operacional, no hypervisor, isolando sistemas inteiros, e não apenas serviços. Tanenbaum trata também das **máquinas virtuais** e dos **hypervisores de Tipo 1 e Tipo 2** na mesma obra, conexão direta com o próximo capítulo.



A imagem resume o problema e a solução: à esquerda, o modelo tradicional — tudo num cômodo só, onde um malware alcança navegador, e-mail, banco e fotos de uma vez; à direita, o Qubes, em que paredes separam cada atividade e o estrago fica preso em um único compartimento.

O mito do antivírus

A resposta tradicional a esse risco é o antivírus, e é importante entender por que ele não conserta o problema de fundo. Um antivírus opera por reconhecimento: ele compara o que está na máquina com uma lista de ameaças conhecidas, ou tenta identificar comportamentos suspeitos. Isso significa, por construção, que ele protege melhor contra o que já é conhecido e pior contra o que é novo — exatamente o tipo de ataque direcionado que ameaça jornalistas e ativistas.

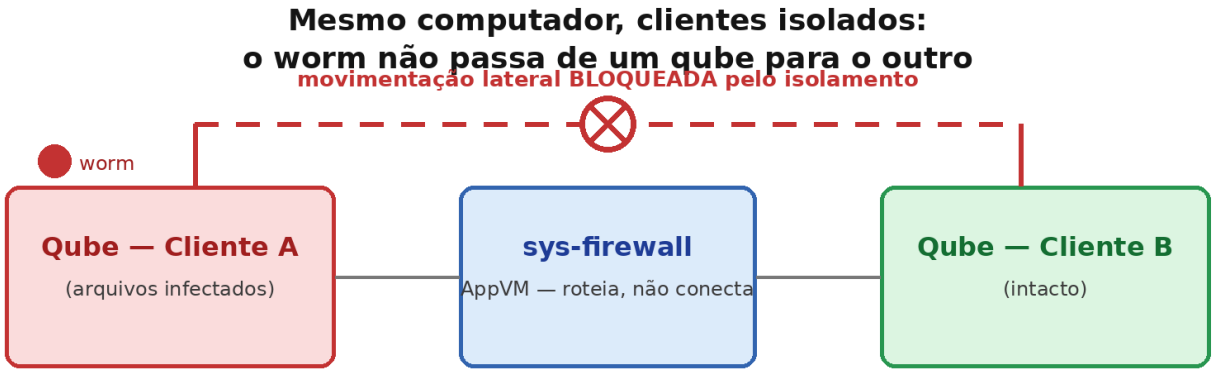
Há um problema mais profundo. O antivírus roda *dentro* do mesmo sistema que ele deveria proteger, com privilégios altíssimos e processando dados não confiáveis para analisá-los. Ele mesmo é um software grande e complexo — e, pela premissa do módulo anterior, contém bugs. Vários ataques sérios já usaram o próprio antivírus como porta de entrada.

Acrescentar um programa grande e privilegiado a um sistema já frágil aumenta a superfície de ataque em vez de reduzi-la.

A lição é mais ampla do que o antivírus: segurança não se resolve com um único produto sobre um único e isolado sistema operacional. Ela precisa ser projetada em **camadas** — a chamada defesa em profundidade —, de modo que, se uma camada ceder, outra ainda contenha o ataque. Empilhar mais um programa privilegiado dentro do mesmo SO não cria camadas; apenas aumenta a superfície. Camadas de verdade são independentes entre si, como o isolamento por hardware, a separação de privilégios e a contenção por compartimento que o Qubes combina.

O Qubes inverte a lógica: em vez de tentar reconhecer e remover o malware depois que ele entrou, ele assume que o malware pode entrar e garante que ele não terá para onde ir. Não é detecção; é contenção estrutural.

Essa contenção tem um efeito prático que vale destacar: a **incapacidade de um worm fazer movimentação lateral** de um qube para outro. Em um sistema tradicional, um malware que entra tende a se espalhar — de um processo a outro e, pela rede, de uma máquina a outra. No Qubes, cada qube é uma fronteira: um worm que infecta um compartimento não encontra caminho até os vizinhos. Imagine um analista de segurança que examina os arquivos de vários clientes, cada cliente em seu próprio qube. Se os arquivos do **Cliente A** estiverem contaminados e o qube em que ele os abre for comprometido, o worm fica preso ali — não pula para o qube do **Cliente B**, nem para o projeto seguinte. O isolamento por hardware barra o acesso direto, e o roteamento de rede (com o firewall no meio) barra o salto pela rede. A figura ilustra: o worm preso no Cliente A, o **sys-firewall** (um AppVM) roteando entre os compartimentos, e o Cliente B intocado.



Um analista de segurança trabalha o material de cada cliente em um qube separado.
Se o qube do Cliente A é comprometido, o do Cliente B permanece intocado.

Quem vem do mundo do desenvolvimento pode perguntar: e os containers, como o Docker? Eles não isolam? Isolam, mas de forma fundamentalmente mais fraca do que máquinas virtuais sobre um hypervisor, e a razão é uma só: containers compartilham o mesmo kernel do hospedeiro.

Um container é um conjunto de processos isolados por mecanismos do próprio kernel — namespaces, cgroups e similares. Funciona bem para empacotar e organizar aplicações, mas, do ponto de vista de segurança, a parede entre dois containers é feita do mesmo kernel monolítico já descrito. Uma falha que permita "escapar" do container leva ao kernel compartilhado e, dali, a todos os outros containers. O isolamento é lógico, não imposto por uma camada menor e independente.

É a diferença entre dividir uma casa com divisórias de gesso e dividir um condomínio com paredes de concreto. O Qubes usa o hypervisor — uma camada pequena, separada e apoiada em hardware — como a parede de concreto entre seus compartimentos. Containers têm seu lugar, mas não são a ferramenta para isolar atividades de confiança radicalmente diferente.

Máquinas virtuais comuns (Tipo 2) não bastam

A intuição seguinte costuma ser: então basta rodar minhas tarefas arriscadas dentro de uma máquina virtual no VirtualBox ou no VMware. É um avanço, mas com um limite estrutural já anunciado no capítulo sobre o Xen. Esses são hypervisors Tipo 2, que rodam *dentro* de um sistema operacional hospedeiro comum. E, como diz a FAQ, uma VM Tipo 2 é apenas tão segura quanto o próprio SO hospedeiro — se o hospedeiro for comprometido, todas as VMs que ele hospeda também ficam, na prática, comprometidas.

Em outras palavras, você empilhou uma camada de isolamento sobre o mesmo kernel monolítico frágil. O hospedeiro — com toda a sua superfície de ataque — continua embaixo de tudo, e comprometê-lo derruba as VMs junto. O Qubes evita isso usando um hypervisor Tipo 1 bare-metal: não há SO hospedeiro gordo embaixo; o atacante precisaria subverter o próprio hypervisor, o que é muito mais difícil.

Computadores separados e o problema do air gap

No outro extremo, há quem use um computador físico para cada tarefa: um só para o banco, outro só para navegar, um isolado da rede (air-gapped) para os segredos. É uma estratégia legítima e, em alguns aspectos, forte: não depende de nenhum hypervisor e complementa a segurança física.

Mas a documentação aponta suas fraquezas sem rodeios. É uma solução pesada e cara. Pior: em geral não há forma segura de transferir dados entre computadores fisicamente separados que rodam sistemas operacionais convencionais — cedo ou tarde você precisa mover um arquivo entre as máquinas, e nesse momento (um pendrive, um cartão) surge o vetor de ataque. E cada uma dessas máquinas, individualmente, continua sendo um sistema monolítico tradicional, vulnerável a tudo o que já foi descrito. Há ainda o detalhe inquietante que a FAQ não omite: já existe, há vários anos, malware capaz de transpor air gaps.

O Qubes oferece a separação forte de várias máquinas, mas em um único equipamento, com um caminho *controlado* e explícito para mover dados entre compartimentos quando você decide fazê-lo — sem pendrives circulando, sem o custo de vários computadores.

Live CDs e sistemas amnésicos

Uma abordagem popular entre quem busca privacidade é o sistema "amnésico" inicializado por pendrive, como o Tails, que não deixa rastros no disco ao desligar. É excelente para o que se propõe, e o próprio Qubes permite rodar o Tails dentro de um qube. Mas, como ferramenta única, ele tem um limite que a FAQ descreve com clareza: um live OS permanece monolítico, no sentido de que todo o software continua rodando no mesmo sistema operacional.

A consequência é direta: se a sua sessão for comprometida, todos os dados e atividades realizados nela também ficam potencialmente comprometidos. Tudo o que você faz na sessão — navegar, abrir um anexo, escrever uma mensagem, acessar uma conta — divide o mesmo espaço. Um único comprometimento durante a sessão alcança tudo o que foi feito nela. A

amnésia protege contra rastros *após* o desligamento, mas não compartimenta o que acontece *durante* o uso.

> *Reforço de privacidade. Para o público deste livro, essa distinção é decisiva. Um jornalista que, na mesma sessão amnésica, abre um documento enviado por uma fonte e depois acessa uma conta identificável pode, com um único anexo malicioso, ligar uma coisa à outra. No Qubes, abrir o documento em um qube descartável e acessar a conta em outro qube mantém as duas atividades separadas — e a amnésia dos descartáveis soma-se à compartimentalização, em vez de substituí-la. Privacidade não é só não deixar rastros; é impedir que atividades que deveriam ser desconexas se contaminem.*

Comparativo do "reasonably secure"

Reunindo o módulo, o quadro fica assim:

- **Windows/macOS/Linux comum:** kernel monolítico, tudo no mesmo espaço de confiança; um comprometimento alcança tudo.
- **Antivírus:** detecção reativa, ele próprio é superfície de ataque; não contém, apenas tenta reconhecer.
- **Containers:** isolamento pelo kernel compartilhado; escapar do container leva ao hospedeiro.
- **VM Tipo 2:** isolamento real, mas apoiado em um SO hospedeiro frágil que continua embaixo.
- **Máquinas separadas:** forte, porém cara, pesada e sem caminho seguro para transferir dados.
- **Live CD amnésico:** sem rastros após desligar, mas monolítico durante a sessão.
- **Qubes OS:** compartimentalização imposta por um hypervisor Tipo 1, com caminhos de transferência controlados e contenção do dano por construção.

É preciso, porém, fechar com o mesmo rigor que o curso adota desde o início — e que a própria documentação adota. O Qubes se descreve como "reasonably secure" (razoavelmente seguro), não perfeito. Ele reconhece que mesmo o código do dom0, por mais simples que seja, pode conter bugs, e que nenhuma ferramenta é infalível. Mais importante para o foco deste livro: o Qubes não promete oferecer propriedades especiais de privacidade em qubes que não sejam Whonix. Isolar não é o mesmo que anonimizar.

> *Reforço de privacidade. Guarde esta frase para o resto do curso: o Qubes entrega **segurança por isolamento** por padrão, mas **anonimato** só onde você o configura explicitamente — essencialmente, através do Whonix sobre Tor. A documentação é categórica ao lembrar que privacidade é muito mais difícil do que se costuma imaginar. Um qube comum esconde suas atividades umas das outras, mas não esconde sua identidade ou localização de um observador de rede. Quem precisa de anonimato — uma fonte, um ativista sob vigilância — precisará dos módulos sobre Whonix, e não apenas do isolamento básico.*

Referências do capítulo

- Qubes OS — FAQ ("reasonably secure"; ausência de garantias de privacidade fora do Whonix). https://doc.qubes-os.org/en/latest/introduction/faq.html
- Qubes OS — Architecture (isolamento por hypervisor). https://doc.qubes-os.org/en/latest/developer/system/architecture.html
- Qubes OS — Introduction. https://doc.qubes-os.org/en/latest/introduction/intro.html

- Andrew S. Tanenbaum — *Sistemas Operacionais Modernos* (estrutura de SO: monolítico, em camadas e micronúcleo/serviços; o "zoológico" de sistemas operacionais; máquinas virtuais e hypervisores; complementar, fora da doc Qubes).

Módulo 3 — O que é o Qubes OS

Depois de entender o problema (Módulos 1 e 2) e a fundação técnica (Módulo 3B, sobre o Xen), é possível enfim definir o objeto do curso com precisão. A documentação oficial abre com uma definição que vale memorizar: o Qubes OS é um sistema operacional livre e de código aberto, orientado à segurança, para computação pessoal de um único usuário. Cada palavra conta: é um sistema operacional completo (não um aplicativo, não uma distribuição "endurecida"), de código aberto, voltado à segurança e pensado para o desktop de uma só pessoa.

O mecanismo que o torna diferente também é dito de forma direta na documentação: o Qubes usa a virtualização baseada em Xen para criar e gerenciar compartimentos isolados chamados *qubes*. É a virtualização do Xen — vista no módulo anterior — posta a serviço do isolamento. Este módulo apresenta esse vocabulário e a filosofia por trás dele; é o mapa conceitual usado no resto do curso.

Segurança por compartimentalização

O princípio central tem nome próprio na documentação. O glossário define o Qubes OS como um sistema cujo princípio principal é a segurança por compartimentalização (isolamento): as atividades são compartimentadas — isoladas — em qubes separados. É a mesma ideia dos Módulos 1 e 2, agora elevada a princípio organizador de todo o sistema: em vez de confiar que cada programa se comportará bem, o Qubes separa as atividades em compartimentos e parte do princípio de que um deles pode cair.

A analogia oficial é a de um navio com cascos estanques. Um casco com divisórias internas não afunda quando uma seção é furada: a água fica contida naquele compartimento. O Qubes aplica essa engenharia ao seu computador. Você não tenta evitar todo furo — isso é impossível —; você garante que um furo não afunde o navio inteiro.

A definição mais enxuta de um qube vem do próprio glossário: um compartimento seguro no Qubes OS. Note um detalhe importante de design: embora hoje os qubes sejam implementados como domínios Xen, a documentação faz questão de afirmar que o Qubes OS independe da tecnologia de compartimentalização que usa por baixo. O qube é um conceito de segurança — um compartimento —, não um detalhe de implementação. Por isso o curso fala em qubes, e não em VMs.

O que é um qube: propósito, natureza e nível de confiança

A Introdução descreve cada qube por três dimensões, e entendê-las é entender como você vai pensar a sua configuração.

A primeira é o **propósito (purpose)**. Cada qube existe para uma finalidade: rodar um conjunto predefinido de uma ou mais aplicações isoladas, para projetos pessoais ou profissionais, ou então cuidar da infraestrutura — gerenciar a pilha de rede, o firewall, ou qualquer outro papel que você definir. Um qube para o banco, um para o trabalho, um para navegação aleatória: cada um com um propósito claro.

A segunda é a **natureza (nature)**. Um qube pode ser uma máquina virtual completa ou enxuta, baseada em sistemas operacionais populares como Fedora, Debian e Windows. É a natureza que define o "sistema" que roda dentro do compartimento.

A terceira — e a mais importante para a sua mentalidade de segurança — é o **nível de confiança (level of trust)**, que vai, nas palavras da documentação, do total ao inexistente. Um qube de confiança total guarda seus segredos; um de confiança nula é onde você abre o anexo suspeito. E como o sistema mostra essa diferença? A Introdução responde: todas as janelas aparecem em um ambiente de desktop unificado, com bordas coloridas não forjáveis, para que os diferentes níveis de segurança sejam facilmente identificáveis. Cada janela carrega uma borda colorida — vermelha, verde, amarela — que identifica, de forma não forjável, a que qube ela pertence.

> *Reforço de privacidade. A borda colorida não forjável é uma defesa direta contra ataques que ameaçam quem precisa de privacidade. Um site malicioso pode imitar perfeitamente a tela do seu banco dentro da janela, mas não pode falsificar a cor da borda imposta pelo sistema. Para um jornalista ou ativista, isso significa poder distinguir, num relance e sem depender da boa-fé do conteúdo, se aquela janela pedindo sua senha pertence ao seu qube confiável ou a um compartimento descartável e hostil. A identidade visual é garantida pelo dom0, fora do alcance do qube comprometido.*

Bordas coloridas não forjáveis: a cor diz a qual qube a janela pertence

A moldura é desenhada pelo dom0 — o conteúdo da janela não consegue falsificá-la.

Na figura, três janelas convivem no mesmo desktop: o qube **trabalho** (verde, confiável), o **pessoal** (amarelo, intermediário) e o **descartável** (vermelho, não confiável). O conteúdo de cada uma poderia ser idêntico — até uma cópia perfeita da tela do seu banco —, mas a **cor da moldura** não muda, porque quem a desenha é o dom0, não o aplicativo. Daí o reflexo a treinar: antes de digitar uma senha, confira a cor da borda.

dom0 e domU: anfitrião e convidados

Dois termos herdados do Xen organizam a hierarquia de privilégio, e o glossário os define com exatidão.

O **dom0** é, segundo o glossário, o primeiro qube iniciado pelo hypervisor Xen no boot. Ele executa a toolstack de gerência do Xen e tem privilégios especiais em relação aos demais domínios, como o acesso direto à maior parte do hardware. É também chamado de domínio anfitrião (host) e classificado como *admin qube* — o qube de administração. Na instalação padrão, o dom0 é o único admin qube e também a interface gráfica do sistema. Ele é a joia da coroa: comprometê-lo é comprometer tudo, razão pela qual, como visto no Módulo 3B, não tem código de rede e você nunca deve rodar nele nada não confiável.

Os **domU** são domínios não privilegiados, também chamados de convidados (guest). O glossário é direto: no Xen, todas as VMs exceto o dom0 são domUs e, por padrão, a maioria dos domUs não tem acesso direto ao hardware. Todo qube que você usa no dia a dia é um domU: sem acesso direto ao hardware, sem privilégio sobre o sistema, isolado pelo hypervisor.

O panorama dos tipos de qube

Aqui está o conjunto de peças que você combinará pelo resto do curso. Cada tipo tem uma definição precisa no glossário.

O **template** é, segundo o glossário, qualquer qube que fornece o seu sistema de arquivos raiz a outros qubes. Ele é o molde: você instala e atualiza software nele, mas — atenção — os templates servem para instalar e atualizar aplicações, não para executá-las. O template não é onde você trabalha; é de onde os outros qubes herdam o seu sistema.

O **app qube** (antigo *AppVM*) é um qube baseado em um template. O detalhe crucial está nessa relação: um app qube não tem sistema de arquivos raiz próprio — toma emprestado o root do template e possui apenas o seu diretório pessoal e os seus arquivos de usuário. Em outras palavras, vários app qubes compartilham o mesmo sistema (vindo do template), mas cada um guarda apenas os próprios arquivos pessoais. É isso que permite ter dezenas de qubes sem desperdiçar disco — e atualizar todos de uma vez, atualizando o template.

O **disposable** (antigo *DisposableVM*) é um qube sem estado: não guarda dado algum entre reinicializações e, a cada início, tem o mesmo sistema de arquivos novo em folha. É o compartimento que nasce limpo e morre sem deixar rastro. Tecnicamente, todo disposable é baseado em um **disposable template**, que define o que estará presente em cada instância descartável.

O **standalone** é qualquer qube que tem o próprio sistema de arquivos raiz e não o compartilha com outro qube — não é baseado em template nem é um template. Útil para casos especiais (um sistema que você instala manualmente), ao custo de ter de atualizá-lo individualmente.

O **service qube** é qualquer app qube cujo propósito principal é prestar serviços a outros qubes; o glossário dá os exemplos canônicos do sys-net e do sys-firewall. São os qubes de infraestrutura que sustentam a rede e o firewall — protagonistas do Módulo 10.

Por fim, um conceito que não é um tipo de qube, mas uma *propriedade*: o **net qube**. O glossário o define como a propriedade de qualquer qube que diz ao sistema qual service qube (se algum) usar para se conectar à rede. E a consequência de segurança mais elegante do Qubes está aqui: se um qube não tem net qube, ele fica offline, desconectado de toda a rede. Tornar um qube totalmente offline é tão simples quanto definir o seu net qube como **None**.

*Reforço de privacidade. Essa propriedade **net qube = None** é a base do vault — um qube sem qualquer conexão de rede, onde você guarda chaves, senhas e documentos sensíveis. Para quem precisa de privacidade, é a garantia mais forte possível contra exfiltração: um segredo guardado em um qube sem rede não pode ser enviado para fora, mesmo que algum software dentro dele seja malicioso, porque simplesmente não há caminho de rede por onde vazar. O vault reaparece no módulo sobre chaves e segredos.*

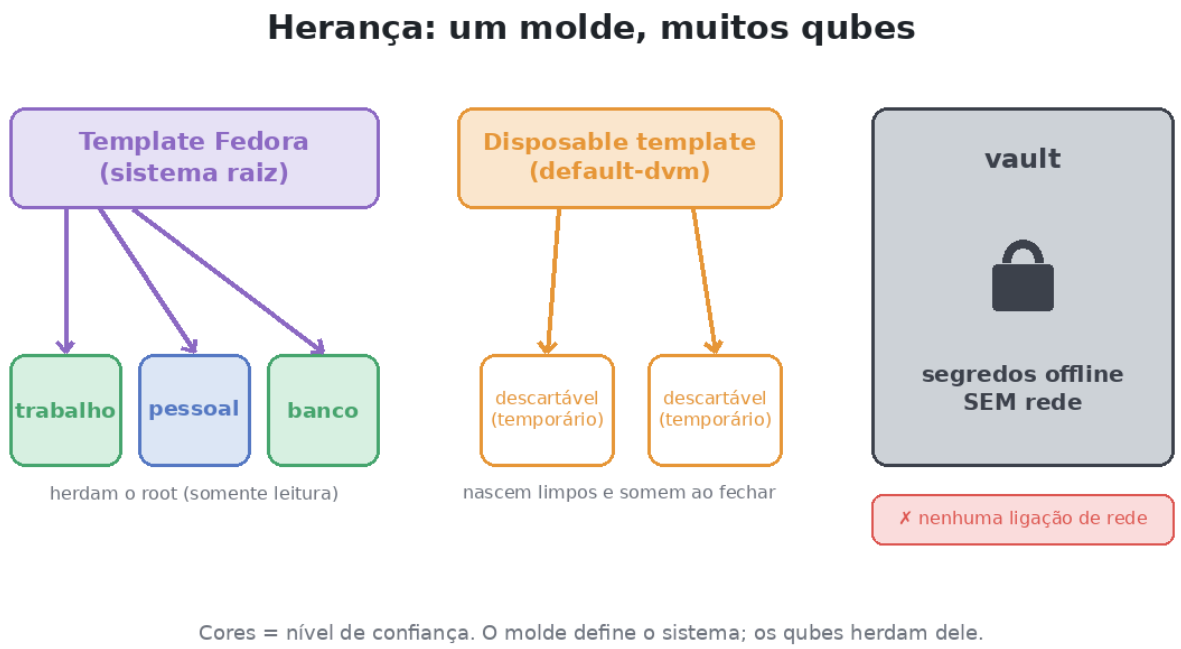


Figura: um template (Fedora) fornece o sistema raiz a vários app qubes; à parte, o disposable template gera descartáveis, e o vault fica isolado, sem rede.

Políticas: o que torna o Qubes especial

Há um último conceito que costura tudo. As interações entre qubes — copiar um arquivo, colar um texto, conceder rede — não acontecem livremente: são governadas por **políticas (policies)**. O glossário as descreve como regras que governam as interações entre qubes, sustentadas pelo sistema qrexec, e arremata sem modéstia que as políticas são uma parte importante do que torna o Qubes OS especial.

Cada política é uma regra aplicada a um qube ou a um conjunto de qubes, que governa como e quando informações ou ativos podem ser compartilhados com outros qubes. O exemplo clássico é a regra que controla como arquivos podem ser copiados entre qubes. Por padrão, essas interações exigem confirmação explícita do usuário — nada flui de um compartimento para outro sem que você autorize. Essa mediação, armazenada em arquivos sob **/etc/qubes/policy.d**, é o que transforma o isolamento bruto do Xen em um sistema utilizável e ainda assim seguro. As políticas reaparecem em detalhe no Módulo 8.

Laboratório 3 — Reconhecendo os tipos de qube

Objetivo: localizar, em um Qubes OS real, os tipos de qube definidos acima, e relacionar cada um à sua definição.

Passo 1 — Abra o **Qube Manager** (menu de aplicativos do dom0). Observe a coluna de tipo/classe e identifique pelo menos um de cada: um template (ex.: **fedora-XX**), app qubes (**personal**, **work**), e os service qubes (**sys-net**, **sys-firewall**, **sys-usb**).

Passo 2 — No terminal do **dom0**, liste os qubes com sua classe e template de origem:

```
qvm-ls --fields NAME,CLASS,TEMPLATE,NETVM,LABEL
```

Repare na coluna **TEMPLATE**: veja como vários app qubes apontam para o mesmo template (herança do root). Repare na coluna **NETVM**: identifique qual qube serve de rede para os outros, e procure algum qube com **NETVM** vazio (offline).

Passo 3 — Verifique o nível de rede de um qube específico e confirme o conceito de net qube:

```
qvm-prefs personal netvm
```

```
qvm-prefs vault netvm
```

(Se não houver um qube **vault**, use qualquer qube e observe o valor.) Um valor **None** confirma um qube desconectado de toda a rede.

Passo 4 — Reflita e anote: para o seu modelo de ameaças (Exercício 1), quais propósitos de qube você já consegue imaginar? Quantos níveis de confiança distintos você precisaria? Guarde — isso vira um desenho concreto no Módulo 18.

> *Atenção: os comandos **qvm-*** rodam no dom0. Apenas leia informações; não altere propriedades de qubes de sistema sem entender o efeito.*

Referências do capítulo

- Qubes OS — Introduction (propósito, natureza, níveis de confiança; bordas coloridas não forjáveis). https://doc.qubes-os.org/en/latest/introduction/intro.html
- Qubes OS — Glossary (template, app qube, disposable, disposable template, standalone, service qube, net qube). https://doc.qubes-os.org/en/latest/user/reference/glossary.html
- Qubes OS — Templates. https://doc.qubes-os.org/en/latest/user/templates/templates.html
- Qubes OS — Command-line tools (qvm-ls, qvm-prefs). https://doc.qubes-os.org/en/latest/user/reference/tools.html

Módulo 3B — O hypervisor Xen por dentro

Tudo o que torna o Qubes OS confiável — o isolamento entre suas atividades, a contenção de um malware dentro de um único qube, a impossibilidade de um navegador comprometido enxergar suas chaves — repousa sobre uma única peça de software: o hypervisor Xen. Se você entende o Xen, você entende *por que* o Qubes funciona, e não apenas *como* clicar nos botões. Por isso este capítulo é mais técnico que os anteriores: ele é a fundação da "prova" de segurança que percorre todo o curso.

A documentação oficial é direta quanto a essa escolha. Segundo a FAQ do projeto, o Xen foi adotado porque a sua arquitetura permite criar sistemas mais seguros — com uma TCB muito menor, o que se traduz em uma superfície de ataque menor. A frase é curta, mas carrega três conceitos a desempacotar: arquitetura, TCB e superfície de ataque.

O que é um hypervisor

Um hypervisor é o programa que cria e isola máquinas virtuais. Ele fica entre o hardware físico (CPU, memória, dispositivos) e os sistemas operacionais convidados, distribuindo a cada um a ilusão de que possui o computador inteiro para si. É o hypervisor que decide qual VM usa a CPU em cada instante, quais páginas de memória pertencem a quem e se uma VM pode ou não falar com a outra.

Existem dois grandes tipos de hypervisor, e a diferença entre eles é o centro do argumento de segurança do Qubes.

Tipo 2 (hospedado)

Um hypervisor Tipo 2 roda *em cima* de um sistema operacional comum. É o caso do VirtualBox e do VMware Workstation: você instala o Windows ou o Linux normalmente e, dentro dele, executa o programa de virtualização. As VMs são, na prática, processos desse sistema operacional hospedeiro.

O problema de segurança é estrutural. Sob uma VM Tipo 2 existe um sistema operacional inteiro — com seu kernel gigantesco, seus drivers, seus serviços de rede — e *todo* esse sistema precisa permanecer íntegro para que o isolamento valha. Comprometer o hospedeiro derruba todas as VMs de uma vez.

Tipo 1 (bare-metal)

Um hypervisor Tipo 1, ou "bare-metal", roda diretamente sobre o hardware, sem nenhum sistema operacional embaixo. O Xen é assim — ele roda direto no hardware nu, ao contrário dos hypervisors Tipo 2, que rodam dentro de um SO hospedeiro.

A consequência de segurança é decisiva. Como explica a FAQ, com um hypervisor bare-metal o atacante precisaria subverter o próprio hypervisor para comprometer o sistema inteiro — o que é muitíssimo mais difícil. Não há um sistema operacional hospedeiro gordo para atacar por baixo: o que separa as VMs é apenas o hypervisor, uma camada deliberadamente pequena.

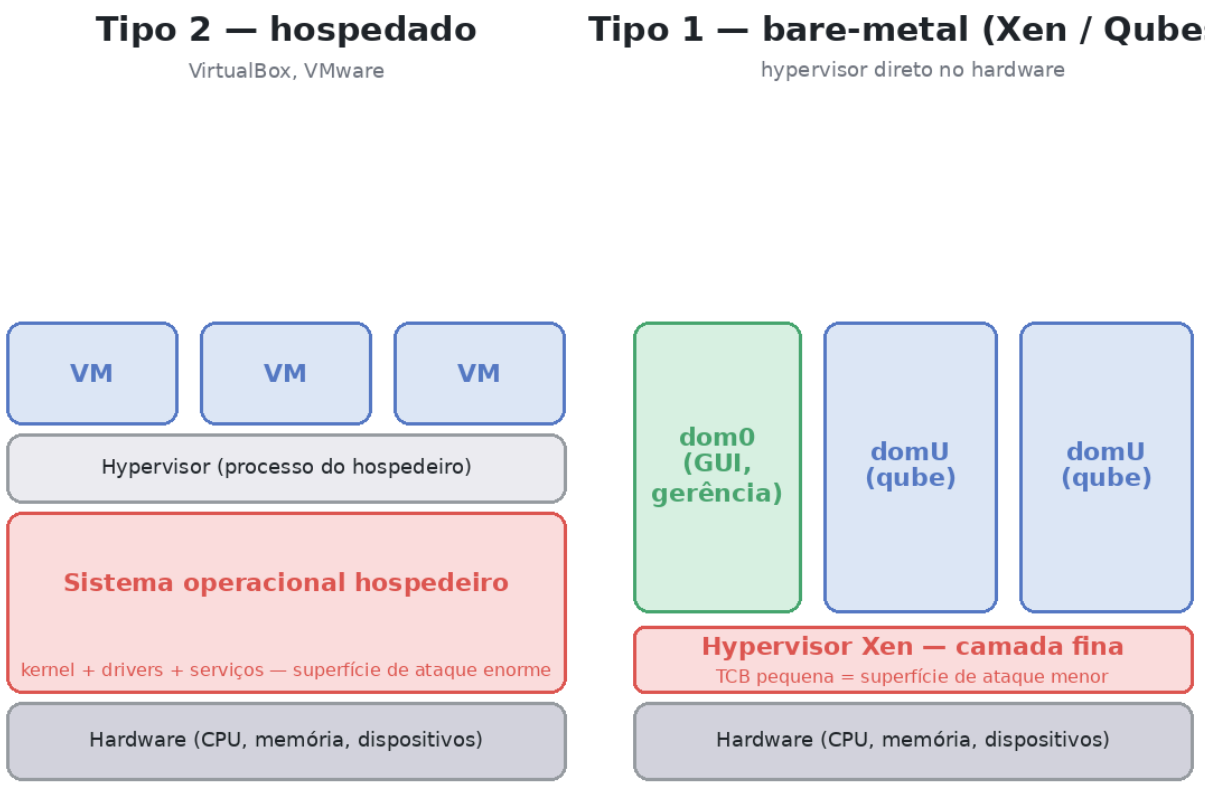


Figura: à esquerda (Tipo 2), todo o sistema hospedeiro (vermelho) precisa estar íntegro; à direita (Tipo 1/Xen), apenas a fina camada do hypervisor.

TCB e superfície de ataque: por que "menor" significa "mais seguro"

Dois termos aparecem o tempo todo quando falamos de Xen, e vale fixá-los porque são a régua da segurança no curso inteiro.

A **TCB (Trusted Computing Base)** é o conjunto de componentes nos quais você é *obrigado* a confiar. Se qualquer parte da TCB for comprometida, toda a segurança do sistema cai. Em um sistema operacional tradicional, a TCB inclui o kernel inteiro, todos os drivers e boa parte dos serviços do sistema — dezenas de milhões de linhas de código.

A **superfície de ataque** é a soma de todos os pontos onde um adversário pode tentar entrar: chamadas de sistema, drivers expostos a dispositivos, parsers que processam dados não confiáveis, interfaces de rede. Quanto mais código exposto, maior a superfície e maior a chance de um bug explorável estar em algum lugar dela.

O argumento do Qubes é puramente quantitativo e, por isso, convincente: o hypervisor Xen é muito menor que um kernel de sistema operacional completo. Menos código na base confiável significa menos lugares onde um bug fatal pode se esconder. É a tradução prática da premissa que abriu este curso — "todo software tem bugs" — em uma decisão de engenharia: se não dá para eliminar os bugs, reduza ao máximo o código em que um bug seria catastrófico.

dom0 e domU: o domínio privilegiado e os convidados

No vocabulário do Xen, cada máquina virtual é um "domínio" (domain). Há dois papéis fundamentais.

O **dom0** (domain zero) é o domínio privilegiado, o primeiro a iniciar. É nele que roda o software de gerência do Xen — a ferramenta de linha de comando que cria, lista e destrói VMs — e, no Qubes, também a interface gráfica do usuário. O dom0 é a parte mais sensível do sistema: comprometê-lo é comprometer tudo.

Os **domU** (unprivileged domains) são todos os demais domínios, os convidados não privilegiados. Cada qube que você usa no dia a dia — trabalho, pessoal, banco, navegação descartável — é um domU. Eles não têm acesso direto ao hardware nem uns aos outros; tudo passa pelo hypervisor, que media e isola.

A decisão de design mais importante do Qubes a respeito do dom0 é o que ele *não* faz. A documentação de arquitetura é explícita: **não há código de rede no domínio privilegiado (dom0)**, e **o código de rede fica isolado (sandboxed) em uma VM sem privilégio, usando IOMMU/VT-d**. Em outras palavras, o componente mais perigoso e mais atacado de qualquer computador moderno — a pilha de rede — é deliberadamente removido da parte mais privilegiada do sistema e jogado para dentro de uma VM descartável e sem privilégio (o **sys-net**). Mesmo que um atacante remoto explore um bug na placa de rede ou no driver, ele cai dentro de uma VM isolada, não no dom0.

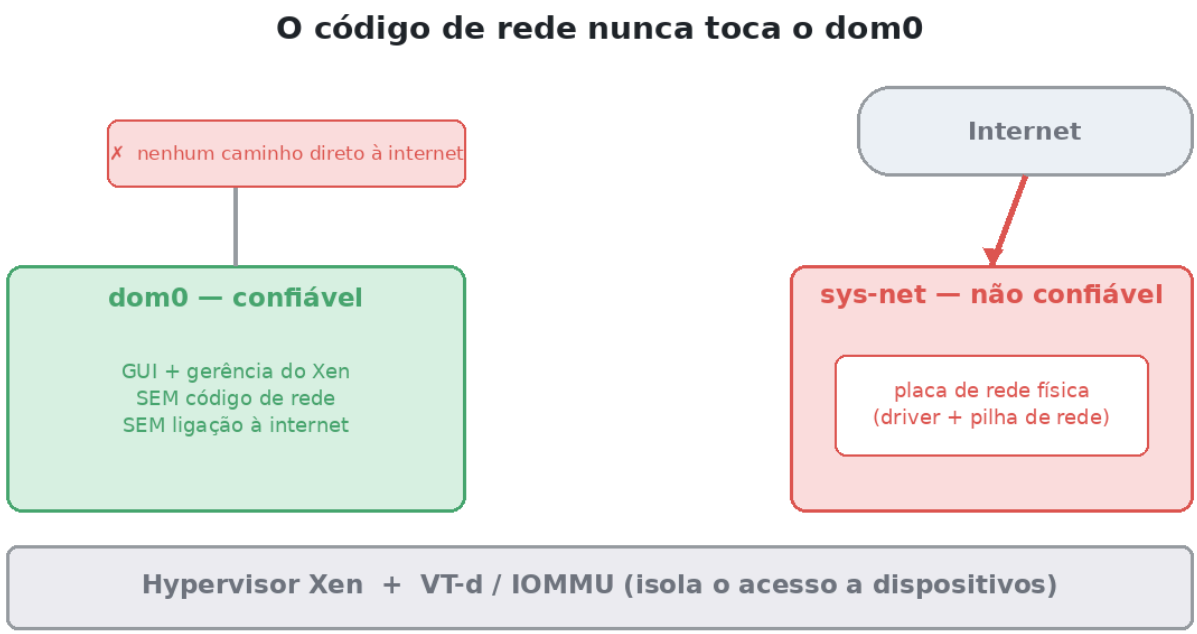


Figura: a internet chega ao sys-net (não confiável), que segura a placa de rede; o dom0 (confiável) não tem código de rede nem caminho direto à internet.

Os modos de virtualização: PV, HVM e PVH

O Xen não virtualiza tudo do mesmo jeito. Ao longo de sua história surgiram três modos, e o Qubes moderno usa cada um com um propósito específico. Entender a diferença ajuda a perceber onde o isolamento é mais forte.

A **paravirtualização (PV)** foi o modo original do Xen. Nela, o sistema convidado "sabe" que está virtualizado e coopera com o hypervisor por meio de chamadas especiais (hypercalls),

sem precisar de suporte de virtualização na CPU. A FAQ resume o mecanismo de separação de privilégios dos domínios PV pelo uso dos anéis 0 e 3 do processador. O PV é elegante e não exige hardware especial, mas expõe uma interface paravirtual ampla entre convidado e hypervisor — historicamente uma fonte de vulnerabilidades. Por isso, no Qubes atual, o PV ficou restrito a um papel coadjuvante: os *stub domains*, vistos adiante.

A **virtualização total por hardware (HVM)** usa as extensões de virtualização da CPU — VT-x (Intel) ou AMD-V, como diz a FAQ — para executar um sistema convidado sem modificações, como se ele estivesse em uma máquina real. É o modo necessário quando o qube precisa de um dispositivo PCI físico (caso do **sys-net** e do **sys-usb**) ou quando se instala um sistema operacional que não foi adaptado para Xen. O HVM tradicionalmente depende do QEMU para emular dispositivos — e aí mora um detalhe de segurança tratado na próxima seção.

A **PVH** é o modo mais moderno e o padrão para a maioria dos qubes no Qubes 4.x. Ela combina o melhor dos dois mundos: usa as extensões de hardware (VT-x e, sobretudo, as tabelas de página estendidas) para isolar CPU e memória de forma robusta, mas dispensa a emulação de dispositivos via QEMU, usando drivers paravirtuais leves para entrada e saída. Menos emulação significa menos código exposto e, portanto, menos superfície de ataque. Por isso a regra prática do Qubes é: qubes sem dispositivo PCI rodam em PVH; qubes com dispositivo PCI rodam em HVM.

> *Aprofundamento (fonte: Xen Project, fora da doc Qubes). O ganho de segurança da PVH vem de eliminar o modelo de dispositivos emulados do plano crítico. Em um HVM clássico, o QEMU emula chipset, BIOS, placas de rede e controladores — milhares de linhas processando entradas do convidado. A PVH troca isso por uma interface mínima de boot e drivers PV, reduzindo drasticamente o que precisa estar correto para o isolamento se manter.*

A virtualização não nasceu com o Qubes. Em *Sistemas Operacionais Modernos*, Tanenbaum dedica um capítulo a ela e formaliza a distinção usada aqui — hypervisores de **Tipo 1**, sobre o hardware, e de **Tipo 2**, sobre um SO hospedeiro. Ele recupera os critérios de **Popek e Goldberg (1974)** para uma arquitetura ser virtualizável e mostra por que o x86 original não os atendia — impasse que abriu dois caminhos: a **virtualização total assistida por hardware** (VT-x/AMD-V, base do HVM) e a **paravirtualização**, em que o convidado é modificado para cooperar com o hypervisor por chamadas explícitas. Foi justamente o **Xen** que popularizou a paravirtualização; ver essa origem explica por que o modo PV existiu e por que o Qubes moderno evoluiu para o PVH.

O papel do hardware: VT-x e VT-d (IOMMU)

A segurança do Xen no Qubes não é só software; ela se apoia em duas funcionalidades da CPU. A documentação as separa de propósito, porque protegem coisas diferentes.

A **VT-x (Intel) ou AMD-V** é a virtualização de CPU e memória. É ela que permite executar convidados isolados com bom desempenho. A FAQ é categórica: o Qubes exige VT-x/AMD-V por usar os modos PVH e HVM, e, sem isso, nenhuma VM inicia numa instalação padrão do Qubes. Junto da VT-x vêm as tabelas de página estendidas (EPT, na Intel), o mecanismo de hardware que dá a cada VM seu próprio espaço de endereçamento físico e impede que uma VM leia ou escreva a memória da outra.

A **VT-d (Intel) ou AMD-Vi / IOMMU** é a virtualização de *dispositivos*. Ela controla o acesso direto à memória (DMA) feito por placas e controladores. Sem ela, um dispositivo malicioso — ou um driver comprometido controlando esse dispositivo — poderia ler qualquer região da memória física por DMA, contornando todo o isolamento de software. A FAQ alerta exatamente para isso: sem VT-d, um atacante poderia executar ataques de DMA para furar as fronteiras de segurança.

Há aqui uma nuance que o curso faz questão de manter: a própria documentação reconhece que todos os demais mecanismos de segurança do Qubes funcionam mesmo sem VT-d. Ou seja, a VT-d é o que torna o isolamento de *dispositivos* realmente sólido (e o que viabiliza separar com segurança o **sys-net** e o **sys-usb**), mas o restante da compartimentalização não depende dela. Saber o que cada peça protege — e o que ela *não* protege — é parte de pensar como um profissional de segurança.

> *Aprofundamento (fonte: Xen Project, fora da doc Qubes). A EPT/SLAT (Second Level Address Translation) adiciona um segundo nível de tradução de endereços em hardware: o convidado traduz endereço virtual para o que ele pensa ser físico, e a EPT traduz isso para o físico real, sob controle do Xen. A IOMMU faz o papel análogo para DMA: cada dispositivo passa a enxergar apenas um espaço de endereços que o Xen lhe autoriza, e não a RAM inteira.*

QEMU fora da TCB: os stub domains

Aqui está um dos pontos em que o Qubes vai além do óbvio. Sistemas de virtualização que usam HVM normalmente executam o QEMU — o emulador de dispositivos — dentro do domínio privilegiado. Como o QEMU processa entradas vindas do convidado, um bug nele poderia ser explorado para escapar diretamente para a parte mais sensível do sistema.

O Qubes recusa esse risco. A documentação afirma que o projeto faz um esforço especial para manter o QEMU *fora* da TCB. O mecanismo é o *stub domain*: para cada HVM, o QEMU correspondente roda dentro de um pequeno domínio PV dedicado e sem privilégio, e não no dom0. Se o QEMU for comprometido, o atacante fica preso nesse stub domain minúsculo, sem alcançar o dom0 nem os outros qubes. É a mesma filosofia de compartimentalização aplicada recursivamente à própria infraestrutura de virtualização.

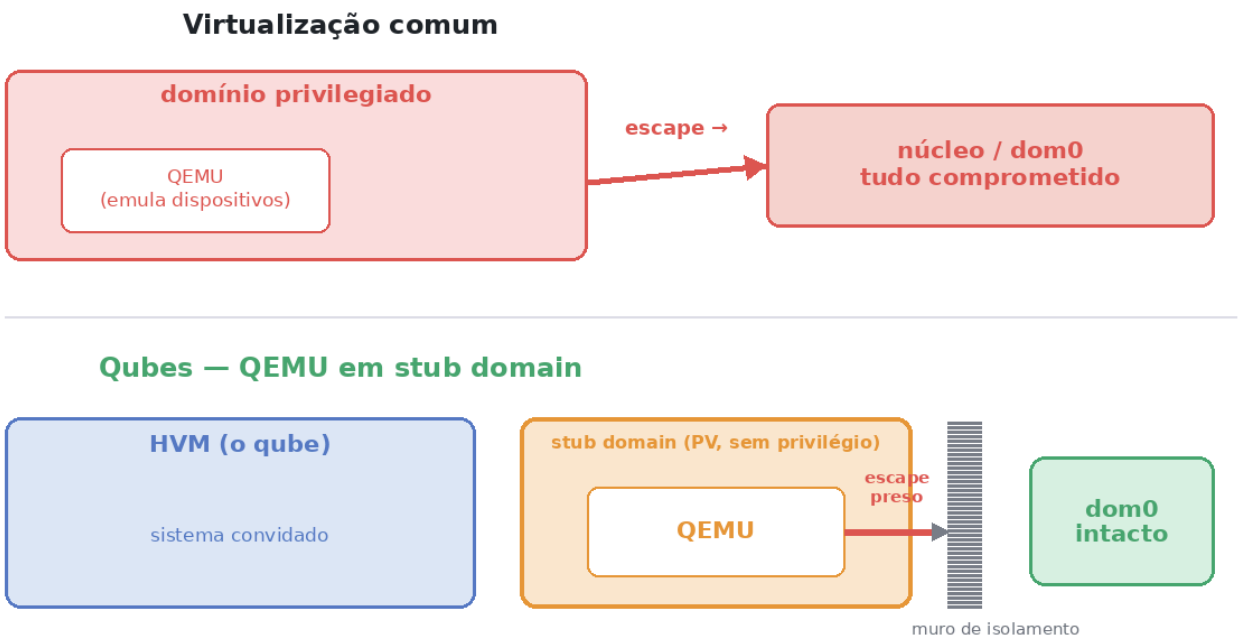


Figura: na virtualização comum, um escape do QEMU atinge o núcleo; no Qubes, o QEMU vive em um stub domain e o escape fica preso, sem alcançar o dom0.

> *Aprofundamento (fonte: Xen Project, fora da doc Qubes). A comunicação controlada entre domínios é feita por duas primitivas do Xen. As grant tables permitem que um domínio conceda a outro acesso a páginas de memória específicas — a base do compartilhamento usado pelos anéis de I/O, sempre explícito e granular, nunca acesso irrestrito. Os event channels são interrupções virtuais que notificam um domínio de forma assíncrona. Sobre essas primitivas mínimas o Xen constrói o **xenstore** (um banco hierárquico de configuração), e o Qubes adiciona sua própria camada (QubesDB e o qrexec) para mediar tudo com política explícita.*

Vale ver o mapa completo dessa integração. O projeto Qubes organiza o sistema em três camadas: na base, a **camada de isolamento** (hoje o Xen; no futuro, possivelmente outros), que fornece memória compartilhada, rede e disco; acima, a **infraestrutura do Qubes** — o canal **vchan** entre domínios, o **QubesDB**, o **qrexec** e o **Qubes GUI** —, que transforma as primitivas cruas do Xen em serviços mediados por política; e, no topo, os **serviços e aplicações** expostos pelo qrexec — de **qubes.FileCopy**, **qubes.OpenInVM** e **qubes.PdfConvert** (que você já encontrou nos módulos anteriores) até a **Admin API** (**admin.vm.***), que gerencia qubes de forma controlada. É essa pilha que faz o Qubes ser mais do que "Xen com várias VMs": um sistema operacional de compartimentos que só conversam pelo que a política permite.

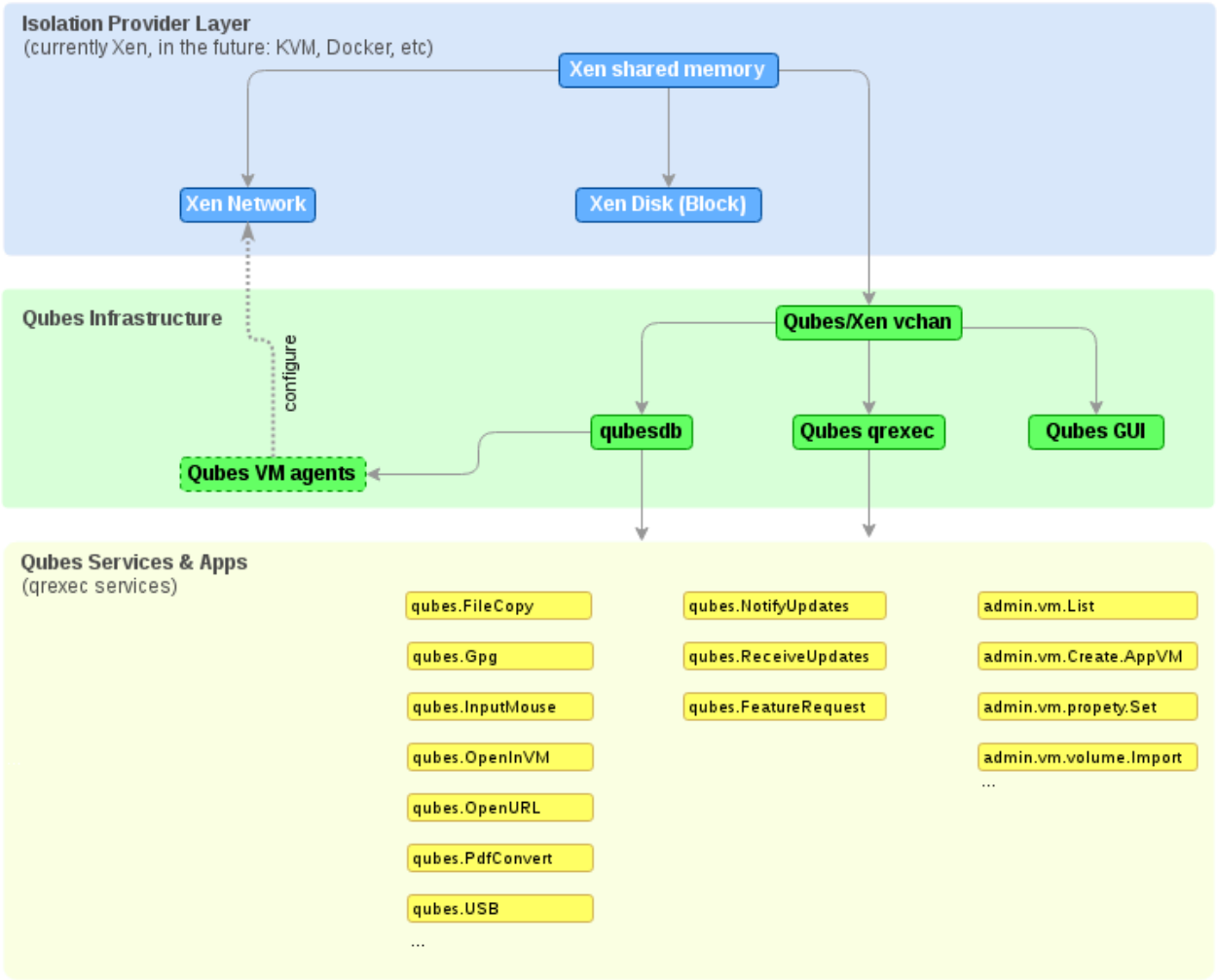


Figura: a pilha de integração do Qubes (fonte: Qubes OS Project) — a camada de isolamento (Xen), a infraestrutura (vchan, QubesDB, qrexec, GUI) e os serviços qrexec / Admin API no topo.

Memória dinâmica: balloon e qmemman

Um detalhe de engenharia explica por que é possível rodar tantos qubes em uma máquina comum: a memória RAM não é dividida em fatias fixas. Em vez de reservar de antemão uma quantidade rígida para cada qube — o que deixaria memória ociosa parada e limitaria o número de compartimentos —, o Xen ajusta a RAM de cada domínio em tempo real, conforme a necessidade do momento. Um qube que está apenas com um editor de texto aberto cede memória para outro que carrega um navegador pesado, e a recebe de volta quando voltar a precisar.

Quem orquestra essa dança no Qubes é o **qmemman** (Qubes Memory Manager), apoiado no *balloon driver* presente em cada convidado. O resultado prático é direto: você cria qubes à vontade e o sistema acomoda o uso real, sem precisar calcular alocações à mão. E o que importa para a segurança é que essa flexibilidade **não enfraquece o isolamento** — cada domínio continua enxergando apenas a memória que o hypervisor lhe concedeu, nunca a dos vizinhos. Elasticidade na quantidade, rigidez na fronteira.

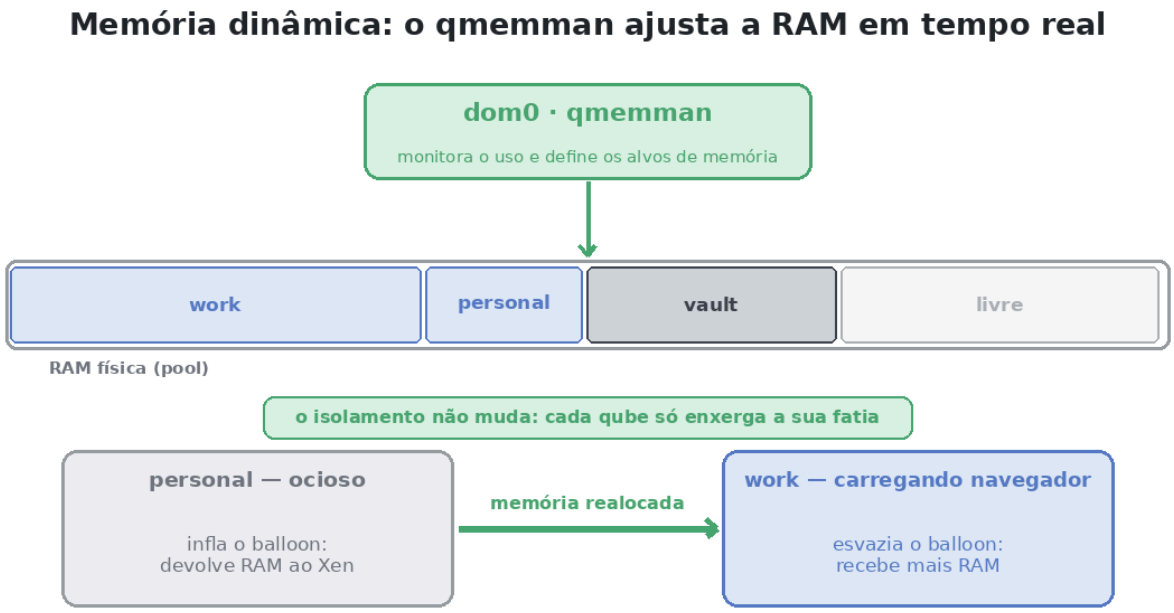


Figura: o qmemman (no dom0) redistribui a RAM física entre os qubes conforme a demanda — um qube ocioso cede memória a outro que precisa —, sem que nenhum domínio enxergue a memória do vizinho.

> *Aprofundamento (fonte: Xen Project + Qubes, complementar). No detalhe, o balloon driver é um módulo dentro de cada convidado que aloca ou libera páginas a pedido do hypervisor: para devolver RAM, ele "infla" — reserva páginas no próprio convidado e as entrega ao Xen, que as repassa a outro domínio; para receber mais, "esvazia". O **qmemman**, rodando no dom0, observa o uso de memória de cada qube (publicado no xenstore) e calcula, em ciclos, novos alvos de alocação, sempre respeitando um mínimo por domínio para que nenhum qube fique sem o essencial. Nada disso cruza a fronteira de isolamento: o balloon opera dentro de cada convidado, o Xen apenas concede ou retoma páginas, e a memória é sanitizada antes de ser reatribuída — um qube jamais lê o que pertenceu a outro.*

XSAs: como o Qubes lida com falhas no próprio Xen

Reduzir a superfície de ataque não é o mesmo que eliminá-la. O Xen, como qualquer software, tem vulnerabilidades — e, por ser a base de toda a segurança do Qubes, uma falha nele é especialmente séria. Por isso o projeto Xen divulga cada vulnerabilidade conhecida como um **XSA (Xen Security Advisory)**, numerado e com a descrição do impacto. A maturidade do Qubes não está em fingir que esses problemas não existem, e sim em manter um **processo claro** para tratar cada um deles.

O fluxo é o seguinte. Quando um XSA é publicado, a equipe do Qubes avalia se ele afeta o sistema e com que gravidade — boa parte dos XSAs atinge recursos ou configurações do Xen que o Qubes sequer utiliza, e por isso não o impacta. Havendo impacto, o projeto emite um **Qubes Security Bulletin (QSB)**: um boletim que explica a falha em linguagem acessível, aponta os qubes afetados e traz instruções precisas ao usuário. A correção é distribuída pelos repositórios oficiais, e aplicá-la costuma ser tão simples quanto atualizar pela ferramenta Qubes Update; alguns QSBs pedem ações adicionais, sempre descritas no

próprio boletim. A FAQ remete ao XSA Tracker, onde se acompanha, advisory por advisory, como cada um afeta — ou não — o Qubes.

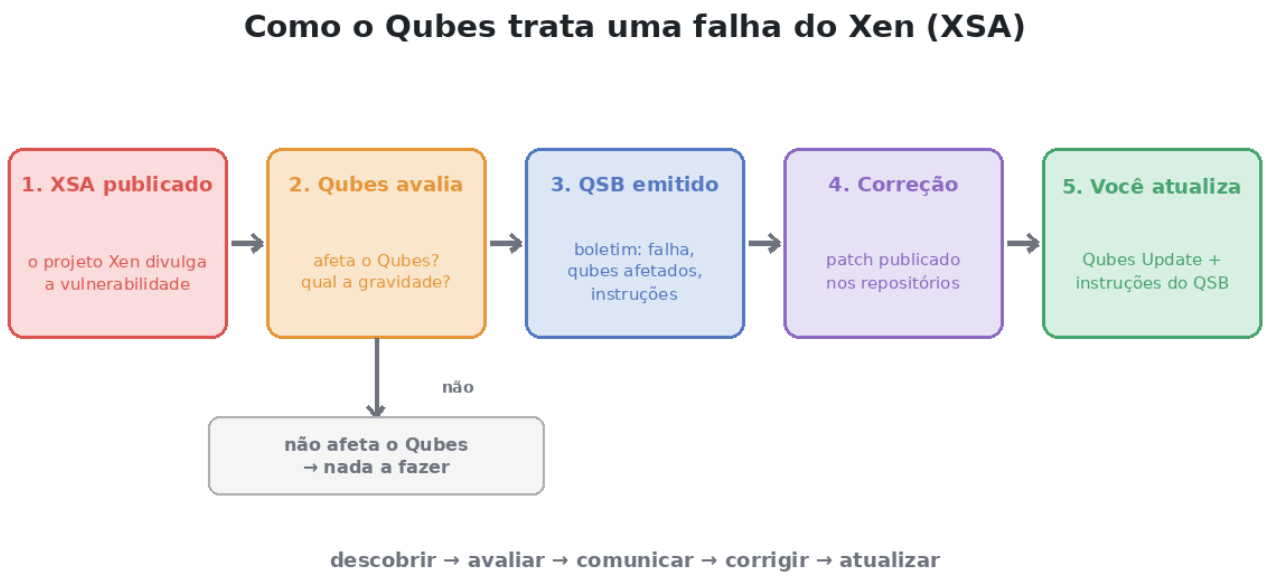


Figura: o ciclo de uma vulnerabilidade do Xen no Qubes — do XSA publicado à atualização aplicada pelo usuário, passando pela avaliação de impacto e pelo QSB.

É esse fluxo que transforma uma má notícia — "há uma falha no hypervisor" — em um procedimento administrável: descobrir, avaliar, comunicar, corrigir e atualizar. A força do Qubes, aqui, não está em alegar perfeição, mas em uma postura realista: a premissa é que falhas existirão, e toda a engenharia — TCB pequena, dom0 sem rede, QEMU em stub domain, isolamento por hardware — serve para que, quando uma aparecer, o estrago seja contido. Para quem opera sob risco, acompanhar os QSBs é parte da rotina (retomado no Módulo 17). O que o Qubes *não* protege é reunido em módulo próprio, mais adiante.

Laboratório 3B — Observando os domínios em execução

Objetivo: ver, com seus próprios olhos, os domínios que o Xen está gerenciando e relacionar a teoria deste capítulo com o sistema real. Execute em uma instalação funcional do Qubes OS.

Passo 1 — Abra um terminal no **dom0** (menu de aplicativos, qube **dom0**, Terminal).

Passo 2 — Liste os domínios diretamente pela ferramenta do Xen:

```
sudo xl list
```

Observe a primeira linha: o **Domain-0**. Repare também nos qubes em execução, cada um deles um domínio separado, com sua coluna de memória (Mem) e estado. Essa lista é a visão do *hypervisor* — é literalmente o Xen mostrando os domínios que ele isola.

Passo 3 — Veja a mesma realidade pela ótica do Qubes:

```
qvm-ls --fields NAME,STATE,CLASS,MEMORY
```

Compare as duas saídas. O **qvm-ls** mostra os qubes com a terminologia do Qubes (AppVM, TemplateVM, etc.); o **xl list** mostra os domínios Xen subjacentes. São duas linguagens para a mesma coisa.

Passo 4 — Confirme que o seu hardware oferece o isolamento de dispositivos discutido no capítulo:

```
sudo xl info | grep -i virt_caps
```

Procure por indicações de **hvm** e **hvm_directio** (este último corresponde à VT-d/IOMMU em uso). A presença de **hvm_directio** confirma que o passthrough seguro de dispositivos PCI — a base para separar o **sys-net** e o **sys-usb** — está ativo na sua máquina.

Passo 5 — Reflita e anote: qual domínio você comprometeria primeiro se quisesse atacar o sistema todo? Por que o dom0 não ter código de rede torna esse ataque muito mais difícil? Guarde sua resposta; ela é retomada no Módulo 3C, quando os cinco mecanismos de isolamento forem reunidos.

> *Atenção: todos os comandos **xl** exigem o dom0. Nunca execute, copie ou cole comandos não confiáveis no terminal do dom0 — ele é a parte mais privilegiada do sistema, e essa disciplina é parte do que mantém o Qubes seguro.*

Referências do capítulo

- Qubes OS — Core3: the next-generation Qubes core stack (diagrama de integração; Admin API; qrexec/QubesDB). https://www.qubes-os.org/news/2017/10/03/core3/
- Qubes OS — FAQ (o QEMU faz parte da TCB?). https://doc.qubes-os.org/en/latest/introduction/faq.html
- Xen Project — Understanding the Virtualization Spectrum (Tipo 1 vs Tipo 2). https://wiki.xenproject.org/wiki/Understanding*the*Virtualization_Spectrum
- Qubes OS — Architecture. https://doc.qubes-os.org/en/latest/developer/system/architecture.html
- Qubes OS — Glossary (dom0, domU, qube). https://doc.qubes-os.org/en/latest/user/reference/glossary.html
- Xen Project — PVH(v2). https://wiki.xenproject.org/wiki/PVH_(v2)
- Qubes OS — System requirements. https://doc.qubes-os.org/en/latest/user/hardware/system-requirements.html
- Xen Project — VTd HowTo (IOMMU / passthrough). https://wiki.xenproject.org/wiki/VTd_HowTo
- Xen Project — Grant Table. https://wiki.xenproject.org/wiki/Grant_Table
- Xen Project — Event Channel Internals. https://wiki.xenproject.org/wiki/Event*Channel*Internals
- Xen Project — XenStore. https://wiki.xenproject.org/wiki/XenStore
- Qubes OS — Qubes memory manager (qmemman). https://doc.qubes-os.org/en/latest/developer/services/qmemman.html
- Xen Project — Memory Management / Ballooning. https://wiki.xenproject.org/wiki/Xen*Project*Software_Overview
- Qubes OS — XSA Tracker. https://www.qubes-os.org/security/xsa/
- Xen Project — Xen Security Advisories (lista oficial). https://xenbits.xen.org/xsa/
- Qubes OS — Command-line tools (qvm-ls, qvm-check). https://doc.qubes-os.org/en/latest/user/reference/tools.html
- Xen Project — xl (toolstack). https://xenbits.xen.org/docs/unstable/man/xl.1.html
- Andrew S. Tanenbaum — *Sistemas Operacionais Modernos* (virtualização: hypervisores Tipo 1/2; critérios de Popek e Goldberg; paravirtualização e Xen; complementar, fora da doc Qubes).

Módulo 3C — Os cinco mecanismos de isolamento

Nos capítulos anteriores ficaram estabelecidos o problema (Módulos 1 e 2), a fundação do hypervisor (3B) e o vocabulário do sistema (3). Este capítulo amarra tudo respondendo à pergunta que move o curso: *por que, concretamente, é possível confiar que um qube comprometido não contamina o resto?* A resposta não é um único truque, mas a soma de cinco mecanismos de isolamento que atuam em camadas diferentes. Quando você entende os cinco, a segurança do Qubes deixa de ser uma promessa e passa a ser uma consequência verificável da arquitetura.

A documentação de arquitetura enuncia o objetivo de forma precisa: o Qubes usa virtualização para isolar programas e até para colocar em sandbox vários componentes de nível de sistema — como os subsistemas de rede e de armazenamento —, de modo que o comprometimento de qualquer um deles não afete a integridade do resto do sistema. Os cinco mecanismos a seguir são os meios pelos quais essa frase se torna verdadeira.

Mecanismo 1 — Isolamento de CPU e memória

O primeiro e mais básico isolamento é o da própria execução: cada qube roda em um domínio Xen separado, com seu próprio processador virtual e seu próprio espaço de memória, e nenhum qube consegue ler ou escrever a memória de outro. Isso não é imposto por software cooperativo — seria frágil —, mas pelo hardware.

Como visto no Módulo 3B, a extensão VT-x/AMD-V e, sobretudo, as tabelas de página estendidas (EPT/SLAT) dão a cada domínio um espaço de endereçamento físico próprio, sob controle exclusivo do hypervisor. Um programa malicioso em um qube pode, no máximo, corromper a si mesmo e ao seu próprio qube; ele não tem sequer um endereço válido para tocar a memória do qube vizinho. É a parede de concreto entre os compartimentos, fundida no silício da CPU.

A consequência prática é direta: comprometer o navegador no seu qube **random** não dá ao atacante nenhum acesso à memória do seu qube **banco** — eles são, do ponto de vista do hardware, máquinas distintas.

Mecanismo 2 — Isolamento de dispositivos

O segundo mecanismo protege contra um vetor que a maioria das pessoas nunca considera: o próprio hardware conectado. A documentação de segurança de dispositivos abre com um princípio que vale gravar: toda capacidade extra que uma VM ganha é superfície de ataque extra — convém sempre atribuir a uma VM apenas o mínimo necessário.

A pilha de rede e a pilha USB — duas das mais complexas e atacadas de qualquer sistema — não vivem no dom0. Como diz a arquitetura, o código de rede fica isolado (sandboxed) em uma VM sem privilégio, usando IOMMU/VT-d, e as pilhas e drivers USB ficam igualmente isolados em VM sem privilégio. É a VT-d/IOMMU que torna isso seguro: ela impede que um dispositivo, via DMA, alcance a memória fora do qube ao qual foi atribuído.

O caso do USB é o exemplo mais didático. A documentação descreve com todas as letras o ataque conhecido como BadUSB: um dispositivo conectado ao dom0 pode atacar um driver USB qualquer, explorar bugs na leitura da tabela de partições ou simplesmente fingir ser um teclado, e cita implementações prontas como o USB Rubber Ducky. A defesa do Qubes é

estrutural: confinar todo o controlador USB em um **sys-usb**, de modo que um pendrive malicioso ataque, no máximo, esse qube descartável — e não o coração do sistema.

Há ainda um cuidado elegante para o caso de dispositivos PCI: por padrão, o Qubes exige que todo dispositivo PCI seja resetável de fora (via hypervisor), o que reinicializa completamente o dispositivo. Assim, um dispositivo que tenha passado por um qube comprometido pode ser reinicializado e voltar a ser confiável, em vez de carregar a contaminação para o próximo qube.

> *Reforço de privacidade. Para quem cruza fronteiras ou opera sob risco, esse mecanismo é concreto: um carregador ou pendrive "emprestado" — vetor clássico contra jornalistas e ativistas — não consegue, num Qubes bem configurado, ir além do **sys-usb**. O controlador físico nunca toca o qube onde estão suas fontes ou suas chaves. O isolamento de dispositivos transforma um ataque físico de "comprometeu tudo" em "comprometeu apenas um compartimento descartável".*

Figura: um BadUSB conectado é contido pelo sys-usb (isolado por VT-d); o ataque não alcança o dom0 nem os demais qubes.

Mecanismo 3 — Isolamento de interface gráfica

Compartimentar a execução não bastaria se todas as janelas dividissem livremente a mesma tela. Por isso o Qubes virtualiza também a interface gráfica. A arquitetura resume os dois objetivos: a virtualização de GUI do Qubes apresenta as aplicações como se estivessem rodando localmente e, ao mesmo tempo, garante o isolamento entre os aplicativos que dividem o mesmo desktop.

Na prática, isso significa que as janelas de todos os qubes convivem em um único desktop — você trabalha de forma fluida —, mas um qube não pode ler o conteúdo da janela de outro, capturar a sua tela inteira, nem registrar o que você digita em outro qube. Não há um servidor gráfico compartilhado onde um aplicativo possa espionar os demais, como acontece em um sistema tradicional. Cada qube só enxerga os pixels que ele mesmo desenha.

A peça visível desse mecanismo é a borda colorida não forjável vista no Módulo 3: ela é desenhada pelo domínio de interface (hoje, o dom0), fora do alcance do qube, e por isso não pode ser falsificada pelo conteúdo da janela.

Reforço de privacidade. O isolamento de GUI neutraliza uma classe inteira de ataques de captura e enganação. Um qube hostil não pode tirar um print da sua janela de mensagens cifradas no qube ao lado, nem registrar a senha que você digita em outro compartimento, nem imitar de forma convincente a janela do seu qube confiável. Para quem protege fontes e comunicações, isso significa que abrir algo perigoso em um qube não expõe o que está acontecendo, na mesma tela, em um qube seguro.

Mecanismo 4 — Isolamento de comunicação (qrexec e políticas)

Isolar não pode significar incomunicabilidade total — você precisa copiar um arquivo, colar um texto, conceder rede a um qube. O quarto mecanismo garante que essas pontes existam, mas sejam estreitas, explícitas e controladas. Toda comunicação entre qubes passa pela infraestrutura **qrexec** e é governada por **políticas**.

Como visto no Módulo 3, o glossário define políticas como regras que governam as interações entre qubes, sustentadas pelo sistema qrexec, e afirma que elas são uma parte importante do que torna o Qubes OS especial. O padrão é restritivo: nada flui de um compartimento para outro sem autorização. Copiar um arquivo entre qubes, por exemplo, exige uma confirmação explícita — não há um sistema de arquivos compartilhado por onde dados escorram silenciosamente.

Sobre essa mesma infraestrutura o Qubes constrói serviços de segurança notáveis, listados na arquitetura: conversores confiáveis de PDF e de imagem, o Split-GPG e proxies USB seguros para dispositivos de entrada (HID). O conversor de PDF confiável é o exemplo perfeito do conceito: em vez de abrir um PDF possivelmente malicioso no qube onde estão seus dados, o arquivo é processado em um qube descartável e devolvido como uma versão segura — a ponte qrexec carrega só o resultado, não o risco.

Reforço de privacidade. O caráter explícito dessas pontes é uma garantia de privacidade por si só. Como nenhuma cópia ocorre sem o seu aval, um malware em um qube não consegue, sozinho, exfiltrar um documento para outro qube com rede e dali para a internet: ele esbarra na política, que exige a sua confirmação. O fluxo de dados entre seus compartimentos é decidido por você, não pelo software — e muito menos pelo atacante.

Mecanismo 5 — Não-persistência

O quinto mecanismo ataca a maior força do malware: sua capacidade de permanecer. Em um sistema tradicional, uma infecção se instala no disco e sobrevive a reinicializações. No Qubes, a arquitetura de templates e descartáveis corrói essa permanência.

Lembre do Módulo 3: um app qube herda seu sistema de arquivos raiz de um template e possui apenas o próprio diretório pessoal e os arquivos de usuário. Isso tem uma consequência de segurança poderosa — o sistema de um app qube vem, a cada inicialização, do template, limpo; mudanças feitas pelo malware no sistema raiz não persistem. E, como a arquitetura destaca, os updates são centralizados: uma atualização feita no template vale para todos os app qubes baseados nele. Você corrige uma vulnerabilidade uma vez, no template, e todos os qubes derivados herdam a correção.

No extremo, está o **disposable**, definido no glossário como um qube sem estado: não guarda dado algum entre reinicializações e, a cada início, tem o mesmo sistema de arquivos novo

em folha. Um disposable nasce limpo, executa a tarefa e é destruído; qualquer coisa que tenha comprometido aquele compartimento simplesmente deixa de existir ao fechá-lo.

> *Reforço de privacidade. A não-persistência é, para o público deste livro, uma das ferramentas mais valiosas. Abrir um anexo enviado por uma fonte desconhecida em um disposable significa que, mesmo que o arquivo seja uma arma, ela não sobrevive ao fechamento do qube — não fica espreitando suas próximas atividades. E como o disposable não guarda estado entre execuções, ele também não acumula um histórico que possa, mais tarde, ligar uma atividade a outra ou a você. Privacidade não é só impedir o vazamento agora; é não deixar um rastro persistente que comprometa você depois.*

Figura: o descartável nasce limpo do disposable template, abre o anexo suspeito e, ao fechar, desaparece sem deixar rastro.

Síntese: cinco camadas, um princípio

Os cinco mecanismos não competem; eles se somam, cada um cobrindo o que os outros não cobrem:

1. **CPU e memória** (VT-x/EPT): um qube não toca a memória de outro.
2. **Dispositivos** (VT-d/IOMMU): hardware malicioso para no qube de sistema, não no dom0.
3. **Interface gráfica** (qubes-gui): um qube não espiona nem falsifica a tela de outro.
4. **Comunicação** (qrexec + políticas): pontes estreitas, explícitas, autorizadas por você.
5. **Não-persistência** (templates e disposables): o malware não permanece.

Juntos, eles transformam a premissa "todo software tem bugs" em um sistema onde um bug explorado fica preso em uma única célula. Esta é a prova prometida no início do curso — não uma alegação de perfeição, mas uma engenharia de contenção em profundidade. O Módulo 19 retoma a outra metade da história: o que esses mecanismos *não* cobrem (canais laterais, falhas no próprio Xen, ataques físicos), e por que reconhecê-lo fortalece, em vez de enfraquecer, a confiança no sistema.

Laboratório 3C — Mapeando os mecanismos no seu sistema

Objetivo: associar cada mecanismo a uma evidência concreta no Qubes OS real, fechando os ganchos deixados nos Laboratórios 3B e 3.

Passo 1 — **Dispositivos.** No terminal do dom0, liste os qubes de sistema e confirme que rede e USB estão isolados em seus próprios qubes:

```
qvm-ls --fields NAME,CLASS,NETVM | grep -E "sys-net|sys-firewall|sys-usb"
```

Passo 2 — **Comunicação (políticas).** Inspecione um arquivo de política e veja as regras que mediam interações entre qubes:

```
ls /etc/qubes/policy.d/
sudo cat /etc/qubes/policy.d/*.policy 2>/dev/null | head -40
```

Identifique pelo menos uma regra que exija confirmação (**ask**) e uma que negue por padrão (**deny**).

Passo 3 — **Não-persistência.** Abra um qube descartável a partir do menu (Disposable), crie um arquivo de teste no /home dele, feche o qube e abra um novo descartável. Confirme que o arquivo não existe mais — o estado não persiste.

Passo 4 — Retome a pergunta do Laboratório 3B (qual domínio você atacaria primeiro?). Agora responda novamente, citando *qual dos cinco mecanismos* impede cada passo do ataque que você imaginou.

> *Atenção: comandos no dom0 são sensíveis. Apenas leia as políticas; não as edite neste laboratório.*

Referências do capítulo

- Qubes OS — Architecture (Qubes Apps sobre qrexec: PDF/Image converters, Split GPG). https://doc.qubes-os.org/en/latest/developer/system/architecture.html
- Qubes OS — FAQ (VT-x/AMD-V; isolamento por hardware). https://doc.qubes-os.org/en/latest/introduction/faq.html
- Qubes OS — Device handling security (atribuir o mínimo a cada VM; BadUSB; reset de PCI). https://doc.qubes-os.org/en/latest/user/security-in-qubes/device-handling-security.html
- Qubes OS — Introduction (bordas coloridas não forjáveis). https://doc.qubes-os.org/en/latest/introduction/intro.html
- Qubes OS — Glossary (disposable: stateless, fresh file system). https://doc.qubes-os.org/en/latest/user/reference/glossary.html
- Qubes OS — How to edit a policy. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-edit-a-policy.html
- Qubes OS — How to use disposables. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-use-disposables.html

Módulo 4 — Hardware, download e verificação da mídia

Sair instalando um sistema operacional de segurança sem pensar no hardware e na procedência da mídia seria como instalar uma fechadura reforçada em uma porta de papelão. Este módulo cobre o que precisa estar resolvido *antes* do primeiro boot: se o seu computador atende aos requisitos, como obter a imagem oficial e — o ponto mais negligenciado e mais importante para o público deste livro — como verificar que essa imagem é autêntica e não foi adulterada no caminho.

Requisitos de hardware

A documentação separa requisitos mínimos e recomendados, e adverte que eles são necessários, mas não suficientes — atendê-los não garante que o Qubes vá instalar e rodar; é o ponto de partida.

No **mínimo** (para executar com extrema lentidão), você precisa de: um processador Intel ou AMD de 64 bits com Intel VT-x com EPT (ou AMD-V com RVI) e Intel VT-d (ou AMD-Vi, também chamado de AMD IOMMU); **6 GB de RAM**; e **32 GB de espaço livre**. Note que os dois recursos de virtualização vistos no Módulo 3B — VT-x/EPT (isolamento de CPU e memória) e VT-d/IOMMU (isolamento de dispositivos) — aparecem aqui como pré-requisitos: sem eles, a "prova" de isolamento não se sustenta.

No **recomendado**, a documentação sobe a régua: um processador Intel recente o bastante para ainda receber atualizações de microcode, **16 GB de RAM**, **128 GB** em SSD, gráficos integrados Intel, um teclado não-USB ou múltiplos controladores USB e um **Trusted Platform Module (TPM)** — este último requerido para o Anti Evil Maid, tema do Módulo 15. A recomendação de teclado não-USB (ou de múltiplos controladores USB) existe para permitir um **sys-usb** que isole o controlador sem prender o seu teclado junto — detalhe que ganha sentido no Módulo 12.

Na prática, esses números têm gradações que valem conhecer. Dá para rodar o Qubes em máquina modesta — inclusive a partir de um pendrive ou SSD externo, num laptop de geração mais antiga com 8 GB de RAM —, mas a experiência é lenta e limita quantos qubes você mantém abertos ao mesmo tempo. Com **16 GB de RAM** e um **SSD**, o uso fica fluido: os compartimentos abrem rápido e vários podem ficar vivos em paralelo. Trate o mínimo como "roda", e o recomendado como "roda bem o dia a dia de vários compartimentos".

Antes de prosseguir, vale consultar a Hardware Compatibility List (HCL) e a lista de hardware certificado: máquinas certificadas passaram por testes formais e oferecem o caminho de menor atrito.

No setup UEFI: habilite VT-x e VT-d

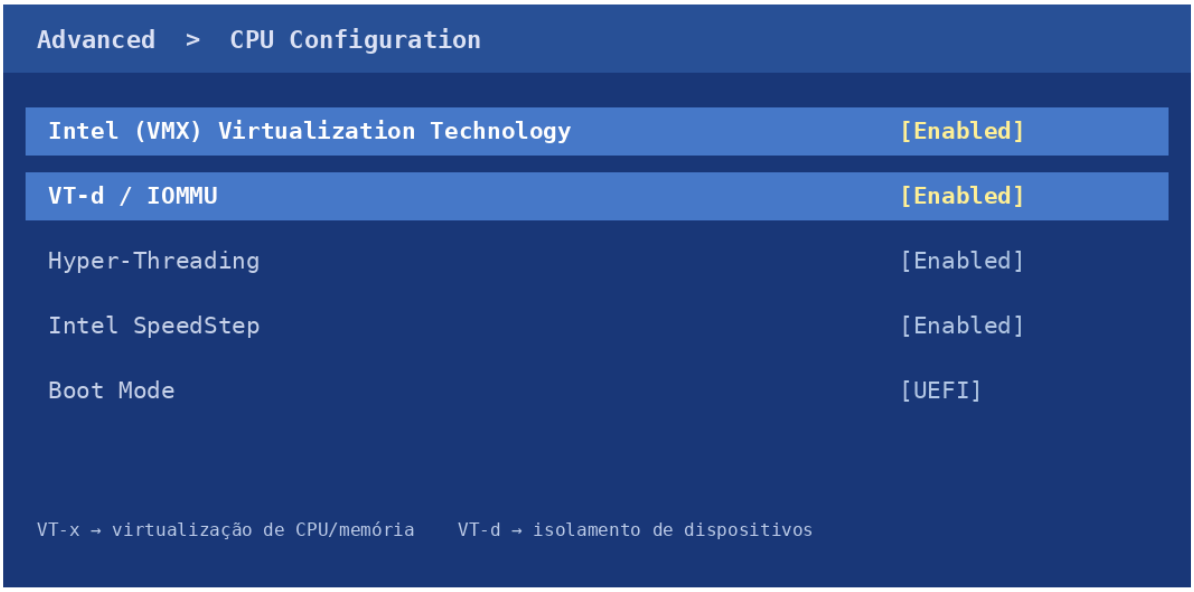


Figura: no setup UEFI, habilite VT-x (virtualização de CPU/memória) e VT-d/IOMMU (isolamento de dispositivos) — base do isolamento do Qubes.

Confiando no seu hardware

Há um limite importante que a documentação faz questão de marcar, e que define a fronteira do que qualquer sistema pode prometer: nenhum sistema operacional, nem mesmo o Qubes, pode ajudar você se a instalação for feita sobre um hardware que já está comprometido. Isso inclui CPUs, GPUs, SSDs, HDDs, a placa-mãe, a BIOS/EFI/UEFI e todo o firmware relevante.

A documentação reconhece que, no mundo atual de ataques de cadeia de suprimentos indetectáveis, não há soluções fáceis. Ferramentas como o Anti Evil Maid ajudam a *manter* a confiabilidade do hardware ao longo do tempo, mas não a estabelecê-la em primeiro lugar. Para quem leva isso a sério, há o caminho do firmware aberto — a doc cita coreboot, Heads e Skulls —, e é justamente o que equipa boa parte do hardware certificado.

> *Reforço de privacidade. Para um jornalista ou ativista, a procedência do equipamento é parte do modelo de ameaças. Um laptop deixado sozinho em um quarto de hotel, recebido de presente ou comprado de forma rastreável pode chegar já comprometido — e, como diz a doc, nenhum software conserta isso depois. É por isso que pessoas sob alto risco preferem hardware com firmware aberto e cadeia de custódia controlada. A privacidade começa antes do sistema operacional: começa em poder confiar na máquina física.*

Baixando a imagem oficial

A imagem do Qubes é distribuída como um arquivo ISO, disponível na página oficial de downloads e em seus espelhos (mirrors). A página oficial — onde fica sempre a lista atualizada de espelhos, com a seção "Mirrors" ao final — é:

https://www.qubes-os.org/downloads/

O servidor primário do próprio projeto é:

https://ftp.qubes-os.org/iso/

Há ainda dezenas de espelhos oficiais espalhados pelo mundo, que servem exatamente o mesmo arquivo. Alguns exemplos estáveis (a lista completa está na página de downloads acima):

- The Linux Kernel Archives (distribuído): https://mirrors.edge.kernel.org/qubes/iso/
- Instituto Superior Técnico, Lisboa (Portugal): https://ftp.rnl.tecnico.ulisboa.pt/pub/qubesos/
- ICM, Universidade de Varsóvia (Polônia): http://ftp.icm.edu.pl/pub/os/qubes/
- dotsrc.org (Dinamarca): https://mirrors.dotsrc.org/qubes/
- Academic Computer Club, Umeå (Suécia): https://mirror.accum.se/mirror/qubes-os.org/

Prefira um espelho próximo de você para acelerar o download; como o arquivo será verificado no passo seguinte, a escolha do espelho não afeta a segurança — só a velocidade.

> ***[TODO Imagem 18]*** *O que mostrar: screenshot da página oficial de downloads (https://www.qubes-os.org/downloads/) com a seção "Mirrors" visível e/ou a listagem de diretório de um espelho exibindo o arquivo .iso. Como produzir: printscreen do navegador (será feito pelo autor).*

Baixar, porém, é só metade do trabalho: um arquivo obtido pela rede pode ter sido substituído por um intermediário malicioso — um servidor comprometido, um espelho adulterado, um ataque na sua conexão. Por isso o passo seguinte, a verificação, não é opcional.

O que uma assinatura digital prova — e o que não prova

Esta é a parte conceitual mais importante do módulo, e a documentação pede explicitamente que mesmo programadores a leiam, porque a maioria das pessoas se confunde com os conceitos básicos por trás das assinaturas digitais.

Uma assinatura digital prova duas coisas, com um grau razoável de certeza: **autenticidade** — que um determinado arquivo foi mesmo criado por quem o assinou — e **integridade** — que o conteúdo do arquivo não foi adulterado. Em outras palavras: que a ISO foi mesmo produzida pelo Projeto Qubes e que ninguém a alterou no caminho.

Mas atenção ao limite, dito com todas as letras: uma assinatura não pode provar, por exemplo, que o arquivo assinado não é malicioso — nada impede alguém de assinar um programa malicioso. A assinatura não diz que o software é bom; diz apenas *quem* o assinou. A decisão de confiar no Projeto Qubes é, nas palavras da doc, mais uma decisão social e política.

Daí nasce a filosofia que dá nome ao método: desconfiar da infraestrutura. Ao verificar a assinatura, você deixa de precisar confiar na rede, no servidor, no provedor, no Wi-Fi — o site qubes-os.org certamente será comprometido um dia, admite a própria documentação. Você passa a confiar apenas em algumas poucas chaves criptográficas que escolheu, e detecta qualquer adulteração no meio do caminho.

> *Reforço de privacidade. "Desconfiar da infraestrutura" é exatamente a postura que protege quem está sendo vigiado. Um adversário capaz de interceptar sua conexão poderia, sem a verificação, entregar uma ISO adulterada — com uma porta dos fundos — e você nunca saberia. A verificação de assinatura transfere a confiança da rede (que você*

> *não controla) para uma chave (que você pode confirmar de várias fontes independentes). É a diferença entre torcer para que ninguém tenha mexido no seu download e ser capaz de provar que ninguém mexeu.*

Todo o esquema se ancora em uma única chave. Os ativos do projeto (ISOs, RPMs) são assinados por chaves de release, e cada chave de release é, por sua vez, assinada pela **Qubes Master Signing Key (QMSK)**. Como diz a doc, a QMSK é a raiz última de confiança do Projeto Qubes OS. A impressão digital (fingerprint) oficial da QMSK é:

```
427F 11FD 0FAA 4B08 0123  F01C DDFA 1A3E 3687 9494
```

O ponto crítico: qualquer um pode criar uma chave que se diz "Qubes Master Signing Key". O que torna a verdadeira verdadeira é a fingerprint. E você não deve confiar em uma única fonte para ela — nem neste material. A documentação recomenda obter a fingerprint de múltiplas fontes independentes, de várias maneiras diferentes (sites, fóruns, redes sociais, slides de palestras, vídeos, até uma camiseta) e comparar caractere a caractere. Se todos baterem, você tem a chave genuína.

Escolhendo a mídia de instalação

Decidida a confiar na ISO verificada, falta gravá-la. A documentação analisa os prós e contras de cada mídia sob a ótica de segurança específica do Qubes. Os pendrives USB são práticos — capacidade flexível, funcionam com **sys-usb** —, mas têm dois contras de segurança: são regraváveis (se montados em uma máquina comprometida depois de gravados, a ISO pode ser maliciosamente alterada) e têm firmware não confiável — o firmware pode ser malicioso mesmo num pendrive novo, e plugar o pendrive em uma máquina comprometida pode comprometer o próprio pendrive (o mesmo princípio do BadUSB do Módulo 3C). Um DVD gravável uma única vez, embora antiquado, elimina o risco de reescrita posterior.

Laboratório 4 — Verificando a ISO do Qubes

Objetivo: baixar e **verificar** a ISO antes de instalar, praticando a filosofia de desconfiar da infraestrutura. Faça em uma máquina com GnuPG (**gpg2**). Substitua **X** pela release que você baixou (ex.: **4.3**).

Passo 1 — Obtenha a Qubes Master Signing Key:

```
gpg2 --fetch-keys https://keys.qubes-os.org/keys/qubes-master-signing-key.asc
```

Passo 2 — Confirme a fingerprint da chave importada e compare-a com a que você obteve de **fontes independentes** (não apenas deste material):

```
gpg2 --fingerprint 0x427F11FD0FAA4B080123F01CDDFA1A3E36879494
```

A linha "Primary key fingerprint" deve ser exatamente **427F 11FD 0FAA 4B08 0123 F01C DDFA 1A3E 3687 9494**. Espaços e maiúsculas não importam; a ordem dos 40 caracteres importa.

Passo 3 — Importe a chave de release e verifique que a QMSK a assinou (procure o prefixo **sig!**, com o **!** indicando assinatura válida):

```
gpg2 --keyserver-options no-self-sigs-only,no-import-clean --fetch-keys
https://keys.qubes-os.org/keys/qubes-release-X-signing-key.asc

gpg2 --check-signatures "Qubes OS Release X Signing Key"
```

Passo 4 — Baixe a assinatura destacada (**.asc**) ao lado da ISO e verifique a ISO:

```
gpg2 -v --verify Qubes-RX-x86_64.iso.asc Qubes-RX-x86_64.iso
```

Uma saída com "Good signature" feita pela chave de release confirma autenticidade e integridade. Qualquer aviso de "BAD signature" significa que a ISO não é confiável — **não a instale**.

Passo 5 — Reflita e anote: por que verificar a assinatura é mais forte do que confiar no cadeado HTTPS do site? (Dica: reveja a ideia de desconfiar da infraestrutura e a frase da doc sobre o site oficial poder, um dia, ser comprometido.)

Referências do capítulo

- Qubes OS — Installation guide. https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/installation-guide.html
- Qubes OS — System requirements. https://doc.qubes-os.org/en/latest/user/hardware/system-requirements.html
- Qubes OS — Certified hardware. https://doc.qubes-os.org/en/latest/user/hardware/certified-hardware/certified-hardware.html
- Qubes OS — Installation security (escolha da mídia de instalação). https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/install-security.html
- Qubes OS — Download Qubes OS. https://www.qubes-os.org/downloads/
- Qubes OS — Download mirrors. https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/download-mirrors.html
- Qubes OS — Verifying signatures (o que a assinatura prova e não prova; desconfiar da infraestrutura). https://doc.qubes-os.org/en/latest/project-security/verifying-signatures.html

Módulo 5 — Instalação passo a passo

Com a ISO verificada (Módulo 4), é hora da instalação propriamente dita. O Qubes usa o instalador Anaconda, o mesmo do Fedora — quem já instalou uma distribuição baseada em RPM vai se sentir em casa. Este módulo percorre o processo do começo ao fim, com atenção especial a duas decisões de segurança que se tomam aqui e que acompanham você para sempre: a **criptografia de disco** e a **escolha dos qubes iniciais**.

Preparando a BIOS/UEFI e dando boot

Antes de tudo, a virtualização precisa estar ligada no firmware. A documentação é categórica: é preciso garantir que a virtualização baseada em IOMMU esteja ativada na BIOS ou UEFI, porque, sem ela, o Qubes OS não conseguirá impor o isolamento. Procure, no setup, as opções **Intel VT-d** (ou **AMD-Vi**) ao lado das extensões **VT-x** (ou **AMD-V**); a doc lembra que, se elas não estiverem na aba "Advanced", podem aparecer na aba "Security".

Em seguida, configure a ordem de boot para iniciar pela mídia onde você gravou a ISO. Em máquinas UEFI, será necessário desabilitar o Secure Boot para o Qubes OS conseguir dar boot. Salve, reinicie e deixe o computador dar boot pelo instalador. Na tela inicial, escolha a opção de **testar a mídia e instalar**. O instalador carrega o Xen logo no início e roda uma checagem de compatibilidade em seguida — se a tela gráfica aparece e a checagem passa, é provável que o Qubes OS funcione no seu sistema.

Um aviso importante da doc: se surgir um alerta sobre IOMMU, não entre em pânico — em geral significa que a virtualização IOMMU não foi ativada na BIOS/UEFI. Sem ela, o sistema perde justamente o isolamento estrito do hardware de rede e de USB, ou seja, o Mecanismo 2 do Módulo 3C.

Criptografia de disco com LUKS

Esta é a decisão de segurança mais importante da instalação, e felizmente o Qubes a toma por você por padrão. Como diz a documentação, por padrão o Qubes OS usa LUKS/dm-crypt para cifrar tudo, exceto a partição **/boot**. Todo o seu sistema — dom0 e todos os qubes — fica cifrado em repouso.

Ao confirmar o destino de instalação, o instalador pede uma passphrase para a criptografia de disco, e ela deve ser complexa. Essa frase secreta é a chave de tudo: sem ela, o disco é um bloco ilegível. E há um aviso que merece ser levado a sério: se você esquecer a passphrase de criptografia, não há como recuperá-la. Não existe "esqueci minha senha" — perdê-la é perder o acesso a todos os dados, definitivamente. Escolha uma passphrase longa, memorável e guardada com cuidado.

> *Reforço de privacidade. A criptografia LUKS é a sua proteção de **dados em repouso** — o cenário do laptop perdido, roubado ou apreendido. Para um jornalista ou ativista, é o que separa "perdi um equipamento" de "expus minhas fontes". Mas guarde desde já uma ressalva aprofundada no Módulo 15: o LUKS protege a máquina **desligada**. Com o computador ligado ou suspenso, a chave está na memória RAM e pode ser extraída por um ataque de cold boot. Por isso, em situações de risco, **desligar** é mais seguro do que apenas suspender. A força de uma passphrase complexa e o hábito de desligar trabalham juntos.*

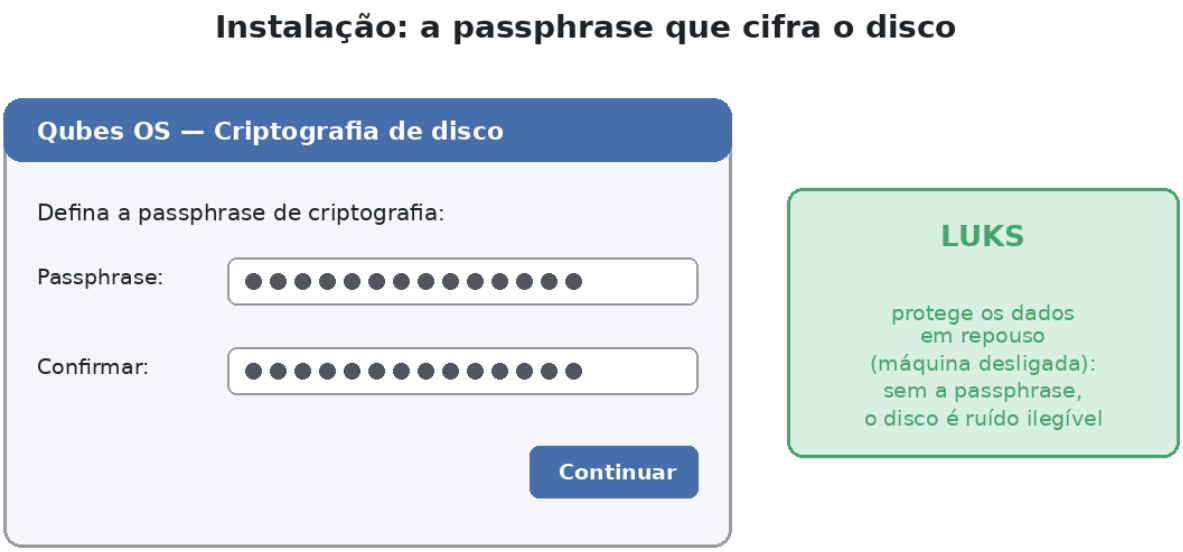


Figura: a passphrase definida no instalador cifra o disco com LUKS; com a máquina desligada, os dados em repouso ficam ilegíveis.

Criação do usuário e início da instalação

Em seguida, você cria sua conta. A documentação esclarece a natureza dela: é uma conta puramente local e offline no dom0; por design, o Qubes OS é um sistema operacional de usuário único, então essa conta é só sua. É a conta com que você fará login após decifrar o disco. Ela tem privilégios de administrador e deve ser protegida por uma senha igualmente complexa. Note que a conta root fica desativada e deve permanecer assim — no Qubes, não se usa uma conta root separada no dom0.

Concluídos os itens obrigatórios, pressione **Begin Installation**. Ao final, ao reiniciar, atenção a um detalhe de segurança que a doc destaca: remova a mídia de instalação e desconecte dispositivos USB desnecessários (dongles, e teclado/mouse USB se não forem os principais). Se você não fizer isso, o segundo estágio da instalação pode tomar decisões por você, rebaixando o seu modelo de segurança — por exemplo, um teclado USB conectado no próximo boot pode fazer o sistema abrir brechas para permitir que esse teclado controle o dom0.

Primeiro boot e configuração dos qubes

No primeiro boot você verá o menu do GRUB e, logo depois, o pedido da passphrase de criptografia. Em seguida vem a etapa que define a "cara" do seu sistema: por padrão, o instalador cria um conjunto de qubes pensados para entregar um ambiente mais pronto para uso desde o início. Vale entender cada grupo, porque várias dessas opções têm impacto direto em privacidade.

Qubes de sistema. A opção de criar os qubes de sistema padrão (sys-net, sys-firewall, DispVM padrão) monta os componentes centrais — sem eles, não há acesso à internet. Há uma sub-opção valiosa: tornar **sys-firewall** e **sys-usb** descartáveis (ligada por padrão), aplicando a não-persistência do Módulo 3C à própria infraestrutura.

Qubes de aplicação. A opção de criar os qubes de aplicação padrão (personal, work, untrusted, vault) monta os compartimentos do dia a dia. A documentação é direta sobre eles:

não há nada de especial nos que o instalador cria — são apenas sugestões que servem à maioria das pessoas. Note a presença do **vault** — o qube sem rede para segredos, ligado ao conceito de **net qube = None** no Módulo 3.

sys-usb e dispositivos de entrada. A opção de criar um **sys-usb** dedicado para segurar os controladores USB. Aqui mora uma decisão de segurança fina: aceitar automaticamente mouse ou teclado USB é desencorajado, porque exigir confirmação no dom0 faz com que um dispositivo malicioso que se apresente como mouse fique inútil até que um diálogo de confirmação no dom0 seja aceito — a defesa contra BadUSB do Módulo 3C, aplicada já na instalação.

Whonix. A opção de criar os qubes Whonix Gateway e Workstation (sys-whonix, anon-whonix) instala a base de anonimato vista no Módulo 11, e ainda permite habilitar as atualizações de sistema e de templates pela rede de anonimato Tor, via Whonix — fazer todas as atualizações passarem por Tor.

> *Reforço de privacidade. Duas opções desta tela são puramente sobre privacidade e merecem sua atenção. Primeiro, tornar o **sys-net** descartável: a doc observa que isso traz um ganho de privacidade, pois o laptop deixa de anunciar a lista de redes Wi-Fi salvas — um rastro que pode revelar onde você esteve. Segundo, atualizar via Tor: impede que um observador de rede saiba quando e o que você atualiza, e dificulta correlacionar seu sistema a você. Para quem está sob vigilância, marcar essas opções na instalação já embute privacidade por padrão.*

Laboratório 5 — Planeje e execute sua instalação

Objetivo: realizar uma instalação consciente, decidindo de antemão as opções de segurança e privacidade. Se você não for instalar em hardware real agora, faça o Passo 1 como exercício de planejamento.

Passo 1 — **Planejamento.** Com base no seu modelo de ameaças (Exercício 1) e nos tipos de qube (Módulo 3), responda por escrito, antes de instalar:

- Qual será a sua passphrase de disco (apenas confirme que é longa e que você consegue memorizá-la) e onde, fisicamente, você a guardará?
- Você marcará a opção de **sys-net descartável** (mais privacidade, perde Wi-Fi automático)?
- Você instalará o **Whonix** e ativará **atualizações via Tor**?
- Você criará um **sys-usb** dedicado e deixará o aceite automático de teclado/mouse USB **desligado**?

Passo 2 — **Execução.** Dê boot pela mídia verificada, ative VT-x/VT-d, desabilite Secure Boot, selecione o disco, defina a passphrase LUKS, crie o usuário e inicie a instalação.

Passo 3 — **Configuração pós-boot.** Na tela de configuração, aplique as decisões do Passo 1. Anote quais qubes o instalador criou.

Passo 4 — **Verificação.** Após o login, abra o Qube Manager e confirme a presença de **sys-net**, **sys-firewall**, **personal**, **work**, **untrusted** e **vault**. Confirme que o **vault** não tem net qube (offline).

> *Atenção: a passphrase de disco não tem recuperação. Não conclua a instalação com uma senha que você não tenha certeza de lembrar.*

Referências do capítulo

- Qubes OS — Installation guide (cifrar tudo menos /boot; passphrase; sem recuperação). https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/installation-guide.html
- Qubes OS — Anti Evil Maid (proteção física complementar). https://doc.qubes-os.org/en/latest/user/security-in-qubes/anti-evil-maid.html
- Qubes OS — How to organize your qubes. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-organize-your-qubes.html

Módulo 6 — Primeiro contato com a interface

Você fez login pela primeira vez. O que vê é um desktop de aparência familiar — o Qubes vem com o XFCE como ambiente de desktop padrão (e também suporta KDE, i3 e AwesomeWM). Mas há uma diferença fundamental sob a superfície: esse ambiente gráfico roda no **dom0**, e o dom0 não é um lugar para trabalhar.

A documentação não poderia ser mais enfática. O dom0 é o qube mestre, com poder máximo sobre tudo o que acontece no Qubes OS, e o mais confiável de todos os qubes. A consequência: se o dom0 fosse comprometido, seria "game over" — um comprometimento efetivo do sistema inteiro. Por isso ele não tem conectividade de rede e serve apenas para rodar o ambiente de desktop e o gerenciador de janelas. A regra de ouro, repetida no curso inteiro: você nunca deve rodar aplicativos de usuário no dom0 — é para isso que existem os seus app qubes.

Internalize isto desde o primeiro minuto: o desktop, o menu, os widgets — tudo isso é a interface do dom0 para você *comandar* os qubes. Seus aplicativos, seus arquivos, sua navegação acontecem *dentro* dos qubes, cada um em sua janela com sua borda colorida. O dom0 é o painel de controle, nunca a oficina.

> *Reforço de privacidade. A ausência de rede no dom0 é, também, uma proteção de privacidade da raiz do sistema. Como o dom0 controla tudo — inclusive as bordas coloridas e o que você digita —, um dom0 com rede seria o alvo perfeito para exfiltrar qualquer coisa. Mantê-lo offline e nunca abrir nele um link, um anexo ou um aplicativo é o hábito mais importante para preservar a confiança de toda a sua compartimentalização.*

O App Menu

O ponto de partida para tudo é o **App Menu** (o ícone "Q"). A documentação o descreve como o lugar onde você abre uma aplicação dentro de um qube, abre um terminal do dom0, acessa ferramentas administrativas como o Qube Manager ou chega aos painéis de configuração.

A diferença em relação a um menu comum é crucial: aqui, cada aplicativo aparece *sob o qube ao qual pertence*. Você não abre "o Firefox"; você abre "o Firefox do qube **personal**" ou "o Firefox do qube **work**". É a interface materializando a compartimentalização: a escolha do qube vem antes da escolha do programa.

Cor e segurança

Esta é a peça de interface mais característica do Qubes. Como visto no Módulo 3, cada qube recebe uma cor, e a cor da moldura de cada janela no desktop corresponde à cor daquele qube. As molduras coloridas, diz a doc, ajudam você a saber qual qube está usando — o que é especialmente útil quando o mesmo programa roda em vários qubes ao mesmo tempo.

A documentação dá o exemplo que todo usuário deve ter na cabeça: se você está logado na conta do banco em um qube enquanto navega aleatoriamente em outro, não vai querer digitar a senha do banco no qube errado por engano — as molduras coloridas ajudam a evitar esse tipo de erro.

Há uma convenção comum — vermelho para o não confiável e perigoso, verde para o seguro e confiável, e amarelo e laranja para o que fica no meio —, mas um detalhe técnico importa:

o próprio sistema não trata as cores de forma diferente; todas são igualmente seguras por si sós. A cor é um auxílio *para você*, humano; ela não confere poder nem fraqueza ao qube. Dois qubes idênticos, um preto e um vermelho, são iguais até você começar a usá-los de forma diferente. Use as cores do jeito que melhor organize a sua cabeça.

> *Reforço de privacidade. Aquela moldura colorida é, na prática, sua linha de defesa contra entregar a informação certa ao compartimento errado. Para quem lida com fontes ou identidades sensíveis, o risco não é só técnico, é humano: digitar uma credencial, colar um endereço ou anexar um documento na janela errada. Como a cor é desenhada pelo dom0 e não pode ser falsificada pelo conteúdo da janela (Módulo 3C), ela é um sinal em que você pode confiar mesmo quando o conteúdo é hostil. Treine o olhar para conferir a cor antes de digitar qualquer coisa sensível.*

Figura: a cor da moldura identifica o qube; confira-a antes de digitar uma senha — verde (banco) não se confunde com vermelho (navegação aleatória).

Os widgets da bandeja

A documentação destaca vários widgets de bandeja que são exclusivos do Qubes OS. Eles são o seu painel de controle do dia a dia, e vale conhecer cada um:

- O **Qubes Clipboard** permite copiar texto facilmente entre qubes e a partir do dom0 — a área de transferência segura, vista a fundo no Módulo 8.
- O **Qubes Devices** permite anexar e desanexar dispositivos — como pendrives e câmeras — aos qubes — o controle de anexar/desanexar do Módulo 12.
- O **Qubes Disk Space** mostra quanto armazenamento você está usando e avisa quando o espaço está acabando.
- O **Qubes Domains** permite gerenciar os qubes em execução, ligá-los e desligá-los e monitorar o uso de RAM e CPU — sua visão rápida do que está rodando.
- O **Qubes Updater** avisa quando há atualizações disponíveis e ajuda a instalá-las — a porta de entrada das atualizações (Módulo 17).

> *Reforço de privacidade. O widget de dispositivos materializa o princípio de que nada é conectado a um qube sem a sua decisão explícita: um pendrive ou uma câmera só passa a existir para um qube quando você o anexa ali. E o widget de clipboard garante que copiar*

e colar entre qubes seja um ato deliberado, e não um vazamento silencioso. Para quem precisa de privacidade, esses widgets são onde você exerce, na prática, o controle sobre o fluxo de dados e dispositivos.

O Qube Manager

Para ver o sistema inteiro de uma vez, há o **Qube Manager**. A doc o descreve de forma simples: para ver todos os seus qubes ao mesmo tempo, use o Qube Manager — ele mostra o estado de todos os qubes do sistema, mesmo os que não estão em execução. Acesse-o pelo App Menu, em Settings → Qubes Tools → Qube Manager.

É no Qube Manager — e no menu — que você cria e remove qubes: a opção "Create New Qube" cria; pelas Settings de um qube você o exclui. O Módulo 8 entra nessas operações em detalhe; por ora, basta saber onde elas vivem.

Comece pelos qubes pré-instalados — e mantenha tudo atualizado

A documentação recomenda começar pelos qubes que o instalador criou: é uma boa ideia começar pelos app qubes pré-instalados — work, personal, untrusted e vault. Quando alguma atividade não se encaixa em nenhum dos seus qubes existentes, você cria um novo — e copia para ele os arquivos de que precisar.

E há um hábito inegociável: é muito importante manter o Qubes atualizado para ter as últimas correções de segurança; atualizar com frequência é uma das melhores formas de se manter seguro contra novas ameaças. O widget Qubes Updater é o seu aliado nisso. O tema é retomado no Módulo 17.

Laboratório 6 — Explorando a interface

Objetivo: reconhecer, na interface real, cada elemento descrito e sentir a compartimentalização na prática.

Passo 1 — Abra o **App Menu** e observe como os aplicativos estão agrupados por qube. Localize o terminal do dom0 e as ferramentas administrativas.

Passo 2 — Abra o **mesmo aplicativo** (por exemplo, um navegador ou um editor de texto) em **dois qubes diferentes** — **personal** e **work**. Coloque as duas janelas lado a lado e observe as **molduras de cores diferentes**. Confirme, lendo a barra de título, a qual qube cada janela pertence.

Passo 3 — Percorra os **widgets da bandeja**: abra o Qubes Domains e veja os qubes em execução com uso de RAM/CPU; abra o Qubes Devices e veja a lista de dispositivos; observe o Qubes Disk Space e o Qubes Updater.

Passo 4 — Abra o **Qube Manager** e identifique os qubes criados na instalação, inclusive os que não estão em execução. Localize **sys-net**, **sys-firewall**, **personal**, **work**, **untrusted** e **vault**.

Passo 5 — Reflita e anote: por que o fato de o desktop rodar no dom0 (sem rede) torna a interface confiável? O que aconteceria à segurança do sistema se você abrisse um navegador no dom0?

Atenção: neste laboratório, apenas explore e abra aplicativos dentro dos qubes. Não execute nada não confiável e jamais rode aplicativos de usuário no dom0.

Referências do capítulo

- Qubes OS — Getting started (tray widgets: Clipboard, Devices, Disk Space, Domains, Updater). https://doc.qubes-os.org/en/latest/introduction/getting-started.html
- Qubes OS — How to copy and paste text. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-copy-and-paste-text.html

Módulo 7 — Conceitos centrais

Há uma única ideia que, uma vez entendida, faz todo o resto do Qubes encaixar: o lugar onde você **instala** software é diferente do lugar onde você **executa** software. A documentação de templates abre exatamente assim — a distinção, no Qubes OS, entre onde você instala o seu software e onde você o executa. Software do dia a dia é instalado nos templates; mas você normalmente trabalha nos app qubes.

Esse capítulo é denso de propósito: ele disseca o modelo de herança e persistência que sustenta a não-persistência (Mecanismo 5 do Módulo 3C) e que você vai usar — consciente ou não — em cada qube que criar. Entender com precisão o que persiste, o que não persiste e por quê é o que separa um usuário que "clica nos botões" de um que sabe onde seus dados realmente estão e onde um malware pode (ou não) se esconder.

TemplateVM: o molde compartilhado

Um **template** é, como definido no Módulo 3, qualquer qube que fornece o seu sistema de arquivos raiz a outros qubes. É o molde de onde os app qubes herdam o seu sistema. A relação tem três propriedades que a documentação destaca, e cada uma é uma vantagem concreta.

A primeira é **segurança**: cada qube tem acesso somente leitura ao template em que se baseia, de modo que, se um qube for comprometido, ele não consegue infectar o template nem os outros qubes baseados nesse template. O acesso é somente leitura — um app qube vê o sistema do template, mas não pode escrevê-lo. Daí a contenção: comprometer um app qube não contamina o molde nem os irmãos.

A segunda é **economia**: dezenas de app qubes compartilham um único sistema raiz, em vez de cada um carregar a sua cópia. A terceira é **velocidade**: criar um app qube novo é extremamente rápido, porque o sistema de arquivos raiz já existe no template. Criar um qube novo é quase instantâneo, porque o sistema já está pronto no template.

Há, porém, uma contrapartida que a doc não esconde e que define um cuidado central: instalar software vai no template, não no app qube. Qualquer software instalado em um app qube (em vez de no template em que ele se baseia) desaparece quando o app qube é desligado. Por isso a recomendação: instale a maior parte do seu software nos templates, não nos app qubes.

AppVM: o que persiste e o que não

Aqui está o coração técnico do módulo, e vale a precisão cirúrgica. Quando um app qube é criado, o conteúdo do diretório **/home** do template-pai não é copiado para o **/home** do app qube filho — o **/home** do app qube nasce independente do **/home** do template.

E, uma vez criado, a regra de persistência é exata: as mudanças em **/home**, **/usr/local** ou **/rw/config** persistem entre reinicializações; nenhuma mudança em qualquer outro diretório de um app qube persiste dessa forma. Ou seja, em um app qube comum:

- Persistem entre reinicializações: **/home** (seus arquivos pessoais), **/usr/local** (software instalado no diretório local) e **/rw/config** (configurações).

- **Não** persistem: todo o resto do sistema de arquivos raiz, que vem do template a cada boot e é descartado ao desligar.

Quer mudar algo fora dessas três áreas de forma permanente? Essa mudança precisa ser feita no template-pai. Um **standalone**, por contraste, não tem template: todo o sistema de arquivos é específico do standalone, e cada mudança feita nele é mantida após o desligamento — flexível, mas sem o benefício de updates centralizados.

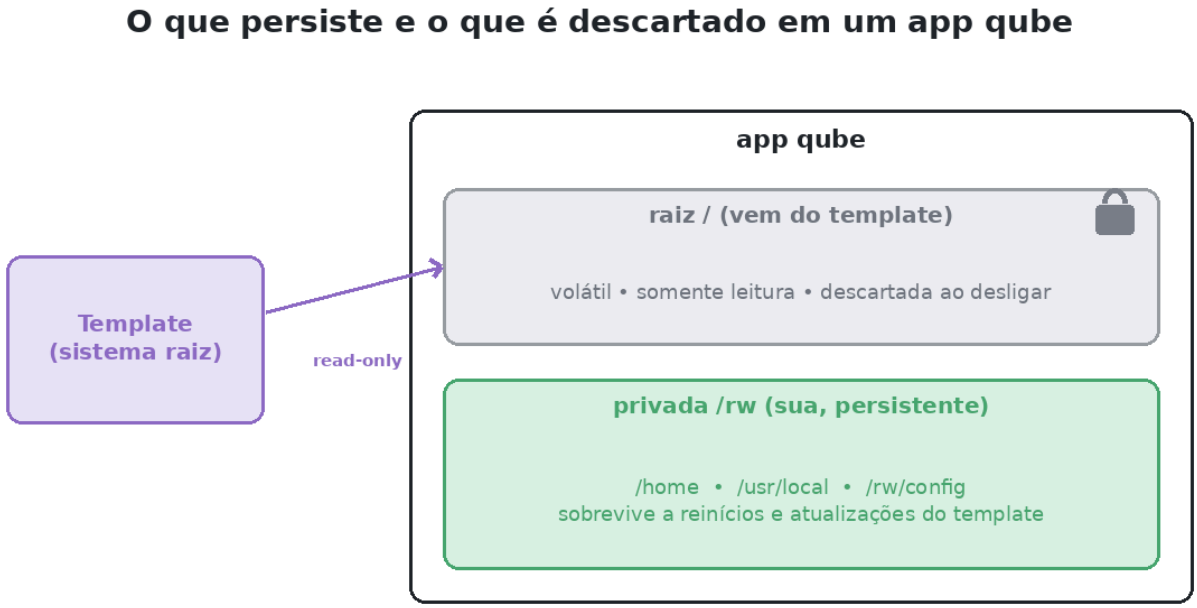


Figura: o app qube combina uma raiz volátil somente-leitura (vinda do template, descartada ao desligar) com uma área privada persistente (/home, /usr/local, /rw/config).

A anatomia do /rw

As três áreas persistentes não estão espalhadas ao acaso: elas vivem em um único volume separado, o **/rw**. A documentação resume: o **/rw** inclui **/home**, **/usr/local** e os bind-dirs. Pense no app qube como a soma de duas partes físicas distintas:

- Um **volume raiz volátil**, fornecido pelo template, montado a cada boot e jogado fora ao desligar. É onde mora o "sistema" — binários, bibliotecas, configurações padrão.
- Um **volume privado persistente** (**/rw**), que é só daquele app qube e sobrevive a reinicializações. É onde mora o "você" — seus arquivos e suas customizações.

Essa separação física é o que torna a não-persistência possível sem destruir seus dados: ao reiniciar, o Qubes troca o volume raiz por uma cópia limpa do template, mas preserva intacto o **/rw**. É também por isso que copiar um arquivo para um qube o coloca, por padrão, em **/home** (dentro do **/rw**): é a parte que fica.

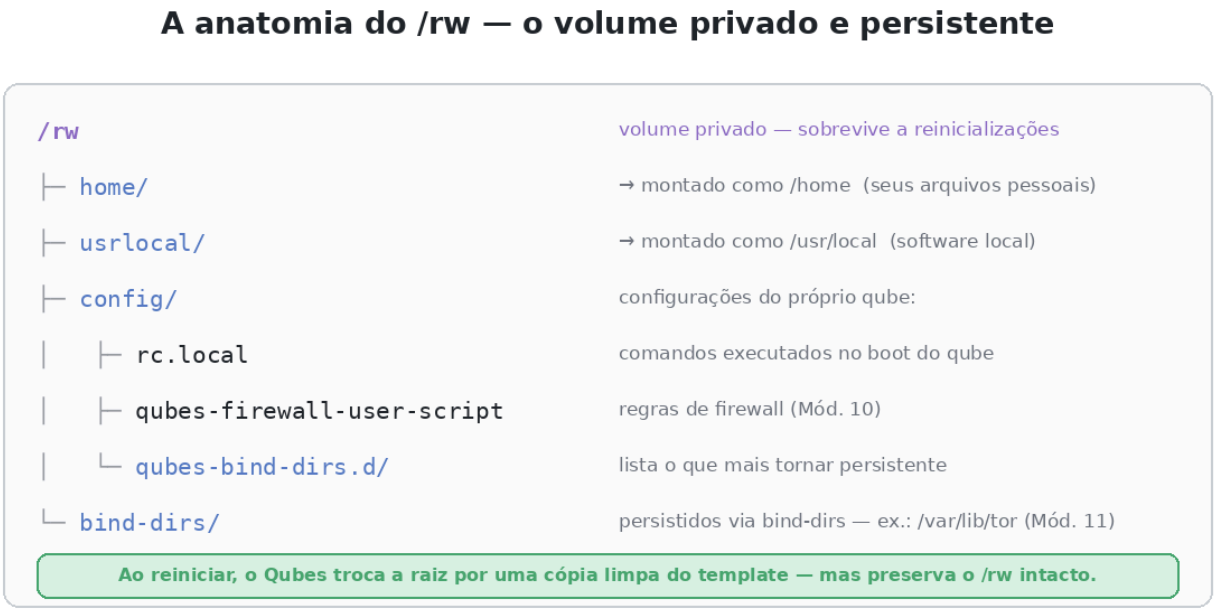


Figura: a estrutura do volume /rw — as áreas que sobrevivem a reinicializações (/home, /usr/local e /rw/config) e os bind-dirs.

bind-dirs: tornando persistente o que normalmente não seria

E quando você precisa que um arquivo *fora* de **/home**, **/usr/local** e **/rw/config** persista, sem transformar o qube em standalone? Para isso existe o **bind-dirs**, um mecanismo pelo qual, nas palavras da doc, quaisquer arquivos ou pastas podem ser tornados persistentes em app qubes — ele oferece um meio de manter, entre reinicializações, arquivos que normalmente viriam do template.

O exemplo da documentação é, convenientemente, de privacidade: no Whonix, o diretório de dados do Tor (**/var/lib/tor**) foi tornado persistente no sys-whonix (uma ProxyVM baseada em template). Assim o **sys-whonix** pode se beneficiar do recurso de anonimato dos "guardas de entrada persistentes" do Tor sem precisar ser um standalone. O bind-dirs é configurado criando arquivos em **/rw/config/qubes-bind-dirs.d/** que listam os caminhos a tornar persistentes.

> *Reforço de privacidade. O caso dos persistent Tor entry guards é um exemplo refinado de como persistência e anonimato se cruzam. Os "guards" são os primeiros relés pelos quais seu tráfego entra na rede Tor; mantê-los estáveis entre reinicializações é uma defesa contra um ataque conhecido em que um adversário tenta, ao longo do tempo, fazer você rotacionar por relés que ele controla para desanonimizá-lo. O bind-dirs permite que o **sys-whonix** preserve essa proteção de privacidade sem abrir mão da segurança dos updates centralizados via template. É a arquitetura do Qubes servindo diretamente ao anonimato.*

Herança de confiança: o template é tão confiável quanto seus filhos

Há uma implicação de segurança da arquitetura de templates que merece um parágrafo próprio, porque é contraintuitiva e importante. Como o template fornece o sistema a todos os seus app qubes, a documentação adverte: o template é tão confiável quanto o app qube mais confiável baseado nele. E a consequência: se o seu template for comprometido — por

exemplo, porque você instalou uma aplicação cujos scripts de instalação eram maliciosos —, todos os seus app qubes baseados nele herdam esse comprometimento.

Em outras palavras: o template é um ponto de confiança concentrado. Um app qube comprometido não infecta o template (acesso somente leitura), mas um *template* comprometido — por exemplo, ao instalar nele um pacote malicioso — propaga o problema para todos os qubes que dele derivam. Daí duas práticas retomadas no Módulo 9: instale no template apenas software de fontes confiáveis, e considere usar templates diferentes para qubes de níveis de confiança muito distintos (por exemplo, um template separado para o que for mais sensível).

Não-persistência como recurso de segurança — e seus limites

A não-persistência do sistema raiz é tentadora de tratar como um antivírus automático: quaisquer mudanças que o qube (ou o malware rodando nele) faça no sistema de arquivos raiz são automaticamente descartadas sempre que o qube é reiniciado. Reiniciou, limpou. Mas a documentação faz uma ressalva importante e crucial, e ignorá-la é perigoso.

A não-persistência vale apenas para o sistema de arquivos raiz, e não para o sistema de arquivos do usuário (onde ficam **/home**, **/rw** e **/usr/local**). E malware esperto sabe disso: é possível que um malware, especialmente um escrito de propósito para atacar o Qubes, instale seus ganchos apenas nos arquivos do diretório pessoal do usuário. A doc dá os exemplos óbvios de esconderijo: o **.bashrc**, o diretório de perfil do Firefox (que guarda extensões), ou documentos PDF/DOC que você abre com frequência. Tudo isso vive em **/home** — e **/home** persiste.

Ou seja: reiniciar um app qube comum **não** garante que ele ficou limpo. A raiz volta nova, mas os ganchos no seu diretório pessoal sobrevivem. A doc aponta o lado positivo — como o malware na raiz fica inativo antes de o sistema de arquivos do usuário ser montado, um usuário habilidoso poderia, em casos especiais, escanear o qube em busca de infecção; e exploits que dependam de bugs no kernel param de funcionar automaticamente assim que o bug é corrigido e a atualização aplicada (via update do template), o que é um recurso excepcional do Qubes OS.

A conclusão prática é direta e leva ao próximo conceito: para tarefas realmente não confiáveis, não confie em "reiniciar o app qube". Use um **disposable**, que não tem sistema de arquivos de usuário persistente e, por isso, inicia completamente "limpo" todas as vezes — limpo significando idêntico ao seu template.

> *Reforço de privacidade. Essa nuance é decisiva para quem abre material de fontes desconhecidas. Se você abre um anexo suspeito em um app qube comum e depois "reinicia para limpar", um malware bem-feito pode ter deixado ganchos no seu **/home** que persistem e continuam espionando suas próximas atividades — exatamente o que você não pode permitir ao proteger uma fonte. A resposta correta é abrir o desconhecido em um **disposable**, onde nada — nem o **/home** — sobrevive ao fechamento. Não-persistência real é privilégio do descartável.*

Os qubes de sistema: sys-net, sys-firewall, sys-usb

Até aqui, o foco esteve nos qubes onde você trabalha. Mas a infraestrutura também é feita de qubes — os **service qubes**, que o glossário define como qualquer app qube cujo propósito principal é prestar serviços a outros qubes, citando o sys-net e o sys-firewall como

exemplos. Eles são app qubes como os outros (baseados em template, com **/rw** próprio), apenas dedicados a um papel de sistema.

- **sys-net** segura os dispositivos físicos de rede (placas Wi-Fi e Ethernet). É a fronteira com o mundo externo e, como visto no Módulo 3B, é onde a perigosa pilha de rede fica confinada, longe do dom0.
- **sys-firewall** fica entre o **sys-net** e seus app qubes, aplicando as regras de firewall por qube (Módulo 10). Como o firewall roda aqui, e não no app qube, malware no app qube não consegue burlá-lo.
- **sys-usb** segura os controladores USB, confinando ataques tipo BadUSB (Módulo 3C). Por padrão, o instalador pode torná-lo descartável.

Note como a topologia padrão encadeia esses qubes: o mundo físico entra pelo **sys-net**, passa pelo filtro do **sys-firewall** e só então chega aos seus app qubes. Cada elo é um compartimento separado, e cada um pode ser descartável.

> *Reforço de privacidade. Tornar **sys-net**, **sys-firewall** e **sys-usb** descartáveis aplica a não-persistência à própria borda do sistema: qualquer comprometimento que entre pela rede ou pelo USB não se acumula entre reinicializações nesses qubes. Para o usuário sob risco, é mais uma camada que impede que um ataque persista silenciosamente no ponto exato em que o sistema toca o mundo externo.*

Laboratório 7 — A anatomia de um qube

Objetivo: comprovar empiricamente o modelo de persistência — o que sobrevive a um reboot e o que não.

Passo 1 — No app qube **personal**, abra um terminal e crie dois arquivos, um em área persistente e outro em área volátil:

```
echo "persistente" > ~/teste-home.txt
sudo bash -c 'echo "volatil" > /etc/teste-root.txt'
```

Passo 2 — Reinicie o qube:

```
qvm-shutdown --wait personal   # (execute no dom0)
```

Depois reabra um terminal no **personal**.

Passo 3 — Verifique o que sobreviveu:

```
cat ~/teste-home.txt        # deve existir (está em /home, dentro de /rw)
cat /etc/teste-root.txt     # NÃO deve existir (raiz voltou do template)
```

Passo 4 — Inspecione o volume persistente e confirme que **/home** e **/usr/local** estão sob **/rw**:

```
ls -la /rw
mount | grep -E "/rw|/home|/usr/local"
```

Passo 5 — Reflita e anote: se você abrisse um anexo suspeito neste qube e depois reiniciasse, por que isso **não** garantiria que o qube ficou limpo? Em qual diretório o malware poderia ter deixado um gancho? Qual seria a escolha correta de qube para abrir esse anexo?

> *Atenção: o **qvm-shutdown** roda no dom0; os demais comandos, dentro do qube. Não instale software em app qubes neste laboratório — lembre que isso não persiste; software vai no template (Módulo 9).*

Referências do capítulo

- Qubes OS — Templates (nota sobre tratar a não-persistência do sistema raiz como recurso de segurança). https://doc.qubes-os.org/en/latest/user/templates/templates.html
- Qubes OS — Glossary (service qube; net qube). https://doc.qubes-os.org/en/latest/user/reference/glossary.html
- Qubes OS — How to make any file persistent (bind-dirs); exemplo do sys-whonix. https://doc.qubes-os.org/en/latest/user/advanced-topics/bind-dirs.html
- Qubes OS — Architecture (rede e USB sandboxed). https://doc.qubes-os.org/en/latest/developer/system/architecture.html

Módulo 8 — Criando e usando qubes

Os capítulos anteriores construíram a teoria; este é onde você põe as mãos. Aqui você vai criar, clonar e remover qubes, e — o ponto mais usado no dia a dia — mover texto e arquivos entre eles de forma segura. Cada operação aqui é uma materialização do Mecanismo 4 do Módulo 3C (comunicação controlada): nada cruza a fronteira entre dois qubes sem a sua autorização explícita. Preste atenção menos aos comandos em si e mais ao *porquê* de cada passo extra: é nele que mora a segurança.

Criar, clonar e remover qubes

Pela interface, criar um qube é direto: no App Menu, "Create New Qube" (em Settings → Qubes Tools); para remover, abra as Settings do qube e use "Delete qube". Ao criar, você define nome, cor (o nível de confiança visual do Módulo 6), template de base e net qube.

Pela linha de comando do dom0, as mesmas operações são rápidas e roteirizáveis:

```
qvm-create --label red --template fedora-XX trabalho   # cria um app qube
qvm-clone personal personal-backup                     # clona um qube existente
qvm-remove trabalho                                    # remove um qube
qvm-prefs trabalho                                     # lista/ajusta propriedades
(template, netvm, memória...)
```

Clonar é especialmente útil para duplicar uma configuração já ajustada, ou para "congelar" um estado antes de um experimento. Lembre do Módulo 7: criar um app qube é quase instantâneo, porque o sistema raiz já existe no template — você só está criando o volume privado **/rw**.

Copiando e colando texto entre qubes

O Qubes tem, nas palavras da doc, uma área de transferência segura entre qubes — o global clipboard — que permite copiar e colar texto puro entre qubes. Ela não é o clipboard comum: tem um passo a mais, de propósito. Para copiar texto do qube A para o qube B:

1. No app do qube A, copie normalmente (**Ctrl+C**).
2. Com o app do qube A ainda em foco, pressione **Ctrl+Shift+C**. Isso leva o texto do clipboard do qube A para o **global clipboard** (uma notificação mostra quantos bytes foram copiados).
3. Selecione o app do qube B e pressione **Ctrl+Shift+V**. Isso copia o texto do global clipboard para o clipboard do qube B e zera o global clipboard, garantindo que só o qube B tenha acesso ao texto copiado.
4. Cole normalmente no qube B (**Ctrl+V**).

Parece elaborado, mas a doc resume o ganho: o método dá a você controle total sobre exatamente qual qube recebe o conteúdo do global clipboard, a cada vez. Detalhe de segurança embutido: não é possível usar o global clipboard para colar de volta no mesmo qube onde o texto foi copiado.

Por que essa dança toda? A documentação dá o exemplo perfeito: sem esse sistema, se você copiasse uma senha do gerenciador de senhas do seu qube vault para outro qube, ela vazaria imediatamente para todos os qubes em execução no sistema, incluindo qubes não confiáveis por padrão, como o sys-net. O global clipboard, ao entregar o conteúdo apenas ao alvo que

você escolheu e depois zerar o global clipboard em seguida, protege a confidencialidade do que você copia.

> *Reforço de privacidade. Para quem manipula segredos — senhas, frases-semente, endereços de contatos —, esse clipboard de passo único é uma defesa direta. Em um sistema comum, qualquer programa rodando pode espiar a área de transferência; um keylogger ou um script malicioso captura a senha que você acabou de copiar. No Qubes, o conteúdo só vai para o qube que você designou, e o global clipboard é zerado em seguida. Você pode reforçar isso ativando o serviço **gui-agent-clipboard-wipe**, que apaga o clipboard do qube de destino um minuto após a colagem — protegendo contra colar acidentalmente uma senha no lugar errado depois.*

A regra de ouro: a direção da cópia importa

Esta é a lição de segurança mais importante do módulo, e ela é contraintuitiva. Transferir dados entre qubes não é simétrico: a direção tem consequências. A documentação adverte que copiar e colar de um qube menos confiável para um mais confiável é sempre potencialmente inseguro, porque o dado copiado pode explorar algum bug hipotético no qube de destino. O exemplo da doc é instrutivo: um link aparentemente inofensivo, copiado de um qube não confiável, pode na verdade ser um grande bloco de lixo que, colado no processador de texto do qube confiável, explora um bug no buffer de desfazer.

Daí a regra de ouro, que vale para texto e para arquivos: copie dados sempre apenas do mais confiável para o menos confiável. Do confiável para o não confiável, sempre. A doc nota que esse problema é geral — vale até para copiar arquivos entre máquinas fisicamente separadas (air-gapped). Internalize: se você precisa levar algo do **untrusted** para o **vault**, pare e pense; você está nadando contra a corrente da segurança.

Há ainda um risco operacional a conhecer: o "focus stealing" (roubo de foco). Como o gerenciador de janelas padrão (Xfce) dá foco a janelas recém-criadas, um qube malicioso poderia "roubar o foco" criando uma janela logo antes de você apertar **Ctrl+Shift+V** e receber o dado em vez do alvo pretendido. A doc sugere mitigar desmarcando, no **xfwm4-settings** do dom0, a opção "Automatically give focus to newly created windows".

> *Reforço de privacidade. A regra de direção é vital para quem protege fontes. Imagine receber um documento por um canal anônimo, abri-lo em um qube descartável e querer "salvar um trecho" no seu qube de trabalho confiável: você está movendo dado de um compartimento hostil para um seguro, justamente o sentido perigoso. A postura correta é processar o conteúdo desconhecido no compartimento descartável (inclusive convertendo PDFs para um formato seguro, Módulo 13) e levar ao qube confiável apenas o mínimo necessário, ciente do risco. Privacidade e segurança aqui se confundem: o caminho dos dados é uma decisão sua, não um automatismo.*

Copiando e movendo arquivos entre qubes

Para arquivos, há dois caminhos. Pelo gerenciador de arquivos: clique com o botão direito no arquivo e escolha "Copy to Other AppVM..." ou "Move to Other AppVM..."; um diálogo **no dom0** pergunta o qube de destino. Pela linha de comando, dentro do qube de origem:

```
qvm-copy arquivo1 arquivo2     # copia para outro qube (pergunta o destino)
qvm-move arquivo               # move (remove da origem após copiar)
```

Em qualquer caso, o arquivo chega ao qube de destino em:

```
/home/user/QubesIncoming/<qube_de_origem>/<arquivo>
```

Esse diretório de "entrada" é uma escolha de segurança: o qube de origem não escolhe *onde* o arquivo cai no destino. A documentação explica por que o sistema é seguro: ele não permite que o qube de origem sobrescreva arquivos arbitrários no qube de destino. Mais ainda, ele não usa nenhum tipo de dispositivo de bloco virtual; em vez disso, usa memória compartilhada do Xen, o que elimina boa parte do processamento de dados não confiáveis. O resultado é notável: o qube que recebe não é forçado a interpretar partições ou sistemas de arquivos não confiáveis — e, por isso, o sistema de cópia de arquivos entre qubes oferece ainda mais segurança do que copiar arquivos entre duas máquinas fisicamente separadas (air-gapped). Copiar um arquivo entre dois qubes é mais seguro do que carregá-lo num pendrive entre dois computadores.

Copiando de e para o dom0 — o caso especial

O dom0 é diferente, e a assimetria reflete sua sensibilidade. Copiar **do** dom0 para um qube é simples, pelo terminal do dom0:

```
qvm-copy-to-vm <qube_destino> <arquivo>
```

O arquivo chega em **/home/user/QubesIncoming/dom0/**. Para texto, há o item "Copy dom0 clipboard" no widget Qubes Clipboard.

Copiar **para** o dom0, porém, é desencorajado com todas as letras: copiar qualquer coisa para dentro do dom0 não é aconselhável, porque isso pode comprometer a segurança do seu sistema Qubes. Por isso, não há um meio simples de copiar nada para o dom0. A razão é a filosofia já vista: o dom0 atua apenas como um "terminal confiável e enxuto", e nenhum aplicativo de usuário roda nele. A doc até desarma a tentação mais comum — copiar uma imagem de papel de parede —: o caminho seguro é exibir a imagem em tela cheia num app qube e tirar um screenshot a partir do dom0. Se for absolutamente necessário, existe o método manual **qvm-run --pass-io <src-vm> 'cat /caminho/arquivo' > /caminho/no/dom0**, mas trate-o como exceção, não como hábito.

> *Reforço de privacidade. A resistência do Qubes a receber dados no dom0 protege a peça mais crítica do sistema de processar conteúdo não confiável — o que, para quem está sob risco, é inegociável. Um único arquivo malicioso aberto no dom0 poderia comprometer todo o seu esquema de compartimentalização e, com ele, todos os segredos que ele protege. A regra é simples e absoluta: dados fluem do dom0 quando necessário, nunca para o dom0.*

Tudo isso é política

Vale fechar conectando com o Módulo 3C: o clipboard global e a cópia de arquivos não são "recursos mágicos" — são **políticas qrexec**, configuráveis como qualquer outra. O global clipboard é configurável como qualquer outra política. Isso significa que um administrador pode, por exemplo, restringir entre quais qubes a cópia é permitida, ou exigir confirmação adicional. As pontes entre compartimentos existem, mas quem decide a largura delas é a política — e, em última instância, você.

A forma como os qubes conversam é um caso clássico de **comunicação por troca de mensagens** e de **chamada de procedimento remoto (RPC)**, como descrevem Tanenbaum e van Steen em *Sistemas Distribuídos*. No RPC, um processo pede um serviço a outro como se chamasse uma função local, mas a requisição e a resposta viajam como mensagens entre

partes isoladas. É o modelo do **qrexec**: um qube pede a outro "copie este arquivo" ou "abra este PDF", e cada pedido é uma mensagem mediada por política — não um acesso direto à memória ou ao disco do vizinho. Esse acoplamento fraco (comunicar-se só por mensagens explícitas, sem memória compartilhada) é o que torna sistemas distribuídos robustos; no Qubes, é o que torna a comunicação entre compartimentos **controlável e auditável**, em vez de um vazamento silencioso.

Laboratório 8 — Transferências na prática

Objetivo: exercitar as transferências seguras e observar onde os dados chegam.

Passo 1 — **Texto.** Abra um editor no **personal**, escreva uma frase, selecione e **Ctrl+C**. Depois **Ctrl+Shift+C**. Vá a um app no **work**, **Ctrl+Shift+V** e **Ctrl+V**. Observe as duas notificações (cópia para o global clipboard e entrega ao destino).

Passo 2 — **Teste a proteção.** Tente colar de volta no mesmo qube de origem com **Ctrl+Shift+V**. Confirme o aviso de que isso é proibido.

Passo 3 — **Arquivo.** No **personal**, crie um arquivo (**echo teste > ~/doc.txt**), use **qvm-copy ~/doc.txt** e escolha **work** como destino. No **work**, confirme que ele chegou em **~/QubesIncoming/personal/doc.txt**.

Passo 4 — **Direção.** Identifique, entre seus qubes, um par "mais confiável → menos confiável" (ex.: **vault → untrusted**) e outro no sentido perigoso. Anote por que uma transferência **untrusted → vault** exigiria cautela extra.

Passo 5 — **Dom0.** No terminal do dom0, copie um arquivo para um qube com **qvm-copy-to-vm personal /etc/hostname** e confirme a chegada em **~/QubesIncoming/dom0/**. Reflita: por que o caminho inverso (copiar para o dom0) é desaconselhado?

> *Atenção: pratique as transferências apenas com dados de teste. A regra de direção (mais confiável → menos confiável) vale mesmo em laboratório.*

Referências do capítulo

- Qubes OS — Getting started (criar/remover qubes). https://doc.qubes-os.org/en/latest/introduction/getting-started.html
- Qubes OS — Command-line tools (qvm-create, qvm-clone, qvm-remove, qvm-prefs). https://doc.qubes-os.org/en/latest/user/reference/tools.html
- Qubes OS — How to copy and paste text (global clipboard; Ctrl+Shift+C/V; wipe; gui-agent-clipboard-wipe). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-copy-and-paste-text.html
- Qubes OS — How to copy and move files (qvm-copy/qvm-move; QubesIncoming; Xen shared memory). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-copy-and-move-files.html
- Qubes OS — How to copy from dom0 (qvm-copy-to-vm; terminal confiável e enxuto; copiar para dom0 desaconselhado). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-copy-from-dom0.html
- Qubes OS — How to edit a policy. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-edit-a-policy.html
- Andrew S. Tanenbaum e Maarten van Steen — *Sistemas Distribuídos: Princípios e Paradigmas* (comunicação por troca de mensagens e RPC; acoplamento fraco; complementar, fora da doc Qubes).

Módulo 9 — Templates: software, atualizações e persistência

No Módulo 7 ficou fixada a distinção entre instalar e executar. Este módulo é a aplicação prática dela: como instalar, atualizar e organizar o software do seu sistema sem quebrar a segurança. A regra de partida vem direto da documentação: quando você quiser instalar software no Qubes OS, deve, em geral, instalá-lo em um template.

Há uma forma elegante de pensar o Qubes que ajuda aqui: ele é, nas palavras da doc, um sistema operacional "meta", capaz de rodar quase qualquer outro SO dentro de si. O Qubes não dita como você instala software — ele apenas fornece o arcabouço onde outros sistemas (Fedora, Debian) rodam. Por isso, instalar um programa em um template Fedora é feito do jeito Fedora; em um template Debian, do jeito Debian. O que o Qubes acrescenta é *onde* e *com que cuidados*.

Instalando software dos repositórios padrão

O fluxo canônico para instalar um programa que está nos repositórios oficiais da distribuição do template tem quatro passos, e a ordem importa:

1. **Inicie o template.**
2. Instale o software do jeito normal daquela distribuição:

```
sudo dnf install <pacote>     # em templates Fedora
sudo apt install <pacote>     # em templates Debian
```

3. **Desligue o template.**
4. **Reinicie todos os qubes baseados nesse template.**

Os passos 3 e 4 não são opcionais. Lembre do Módulo 7: os app qubes herdam o sistema raiz do template a cada boot. O software recém-instalado só aparece nos app qubes depois que eles reiniciam e pegam a nova versão do template. Esse é também o motivo de você instalar uma vez e o programa ficar disponível em todos os qubes daquele template — a economia vista no modelo de herança.

> *Nota importante: ao contrário de atualizar (visto adiante), instalar pacotes com os comandos diretos do gerenciador (**dnf install**, **apt install**) dentro do template é o procedimento recomendado e correto.*

Um exemplo concreto do poder de configurar **no template** para valer em todos os filhos é o endurecimento do navegador. Ajustes feitos na interface do Firefox de um app qube **não** se propagam, porque ficam em **~/.mozilla** (no **/home**, que é por-qube). Já uma política empresarial vale para todos: coloque um arquivo **policies.json** (desabilitando telemetria, pedidos de localização, atualizações automáticas do próprio Firefox, etc.) em **/usr/lib/firefox-esr/distribution/** dentro do template; desligue o template e reinicie os qubes, e a política passa a valer em cada app qube baseado nele. Confira abrindo **about:policies** no navegador do qube. Guarde o **policies.json** no vault para editar com calma e distribua entre qubes com o **qvm-copy**.

Software de fontes externas e o risco de dar rede ao template

Nem todo software está nos repositórios padrão. Para esses casos há caminhos mais cuidadosos. Em templates Debian, o Qubes traz o **extrepo** pré-instalado para gerenciar repositórios externos com segurança, sem precisar dar rede ao template. Mas há um alerta de segurança que vale a pena destacar, porque toca o ponto de confiança do Módulo 7: a recomendação é instalar software de terceiros em um template dedicado, e não no template padrão.

E quando o método exige dar acesso de rede direto ao template? A documentação é enfática: esse método dá ao template acesso direto à rede, o que é arriscado e não é recomendado para templates confiáveis. Por padrão, templates não têm rede normal justamente porque um template é um ponto de confiança concentrado: comprometê-lo contamina todos os seus filhos.

> *Reforço de privacidade. A recomendação de usar um **template dedicado** para software de terceiros é, na prática, compartimentalização aplicada à confiança do seu próprio software. Para quem depende do sistema para proteger fontes, instalar um aplicativo de procedência duvidosa no mesmo template do qube de trabalho seria estender a confiança a algo que não a merece. Manter um template separado para o que é menos confiável garante que, se aquele software trouxer um problema, ele fique restrito aos qubes daquele template — e não alcance os compartimentos onde estão seus dados sensíveis.*

Escolhendo o template: Fedora, Debian e minimal

O Qubes vem com templates prontos, e a escolha tem implicações. **Fedora** e **Debian** são as opções padrão, completas e adequadas à grande maioria dos usuários — a diferença entre elas é mais de preferência e ciclo de lançamento do que de segurança.

Há também os **templates minimal**, que a documentação descreve como versões enxutas: têm apenas os pacotes mais essenciais instalados. O ganho relevante está na sequência: os templates minimal consomem menos recursos, reduzem a superfície de ataque e permitem uma compartimentalização mais fina. Menos pacotes significa menos código, logo menos superfície de ataque — o princípio do Módulo 1 aplicado ao template. O custo é o esforço: a doc avisa que eles são destinados apenas a usuários avançados e que a maioria das coisas não funciona de imediato. Não comece por eles; chegue a eles quando dominar o básico.

> *Reforço de privacidade. Para um perfil de alto risco, o template minimal é uma ferramenta poderosa: você constrói um sistema que contém exatamente o necessário para uma tarefa e nada além disso. Um qube de comunicação segura, por exemplo, pode ser baseado em um template minimal com apenas o aplicativo de mensagens e suas dependências — sem navegador, sem leitor de PDF, sem dezenas de serviços que ampliariam a superfície de ataque. Menos software é menos coisa que pode falhar e menos coisa que pode vazar.*

Atualizando com segurança

Atualizar é, nas palavras da doc, uma parte extremamente importante de manter a sua instalação do Qubes segura. Quando há um incidente sério, o projeto emite um **Qubes Security Bulletin (QSB)** pelo Qubes Security Pack, e é muito importante ler cada novo QSB e seguir as instruções ao usuário que ele contiver.

A ferramenta correta para atualizar é a **Qubes Update**, que aparece como um ícone na área de notificação quando há atualizações disponíveis. Ela cuida do dom0 e dos templates de

forma centralizada e segura. Pela linha de comando do dom0, os equivalentes são **qubes-dom0-update** (para o dom0) e **qubes-vm-update** (para os domUs).

Aqui está uma diferença crucial em relação a instalar: a documentação avisa que atualizar com comandos diretos como **dnf update** e **apt update** não é recomendado, pois eles contornam as proteções de atualização embutidas no Qubes OS. Ou seja — instalar com **dnf install** no template é correto; mas *atualizar* com **dnf update** direto, não: isso contorna as proteções de atualização do Qubes (como o updates proxy e a verificação). Use sempre a Qubes Update.

Dois cuidados operacionais que a doc destaca: o dom0 deve ser reiniciado após qualquer atualização de Xen e de kernel; e, após atualizar um template, primeiro desligue o template e depois reinicie todos os qubes em execução baseados nele — a ferramenta tenta fazer isso por você, mas salve seu trabalho antes.

> *Reforço de privacidade. A doc menciona, quase de passagem, algo de enorme valor para quem precisa de anonimato: você pode baixar as atualizações por Tor, via **sys-whonix** (a doc só ressalva que isso pode demorar bastante). Atualizar via Tor impede que um observador de rede saiba quando você atualiza, quais pacotes baixa e, por correlação, qual sistema e versão você usa — informação que poderia ajudar a identificar você ou a planejar um ataque direcionado. Para o usuário sob vigilância, vale a espera. (O anonimato de rede é configurado no Módulo 11.)*

Standalones: quando você abre mão do template

Por fim, lembre da alternativa do Módulo 3: o **standalone**, um qube com sistema raiz próprio, não baseado em template. Ele é útil quando você precisa de um sistema que muda de forma persistente (por exemplo, um ambiente de desenvolvimento muito específico, ou um sistema operacional instalado manualmente). O custo, já sabido, é que você perde os updates centralizados: cada standalone precisa ser atualizado individualmente, e cada um é uma superfície de manutenção separada. Use-os com parcimônia, quando o modelo template/app qube realmente não servir.

Rodando qualquer sistema como um qube (HVM standalone)

Além dos app qubes baseados em template, o Qubes pode rodar **um sistema operacional qualquer** a partir de uma ISO, isolado do resto — útil para testar uma distribuição, usar um SO específico ou conter um ambiente que você não quer misturar. Cria-se um qube do tipo **standalone** (sistema próprio, persistente, sem template), **template = nenhum** e modo de virtualização **HVM** (o modo de hardware completo do Módulo 3B, necessário para bootar um SO não adaptado ao Xen), apontando a ISO como mídia de inicialização.

Dois tropeços comuns valem o aviso, porque derrubam a instalação logo no começo: **memória inicial baixa** — a RAM efetivamente disponível fica um pouco abaixo da que você define, então, se o instalador exige 1 GB e você reserva exatamente isso, ele pode reportar menos e recusar; reserve folga (por exemplo, 1,5 GB de RAM inicial) e deixe o máximo maior, pois o qube só cresce conforme a demanda (Módulo 3B). E **disco insuficiente** — dê espaço de sistema e privado suficientes para o SO caber, ou a instalação falha.

Um detalhe de rede que confunde muita gente: um SO não-Qubes dentro do HVM **não recebe rede automaticamente** (não há DHCP na rede ponto a ponto do Qubes). É preciso configurar **IPv4 manual** no convidado com os valores que o Qubes atribuiu ao qube — que

você lê, no próprio qube, com **qubesdb-read /qubes-ip**, **qubesdb-read /qubes-gateway** e **qubesdb-read /qubes-primary-dns**; a máscara é **255.255.255.255** (o enlace /32 do Módulo 10). Preencha endereço, máscara, gateway e DNS com esses valores e reconecte a interface.

> *Reforço de privacidade. Rodar uma ISO em um standalone **não a torna mais segura** — ela é tão frágil quanto seria em qualquer lugar. O ganho é a **contenção**: se aquele sistema for comprometido, o estrago fica preso naquele qube, sem alcançar o resto da sua vida digital. É o isolamento do Qubes servindo até ao que você instala por fora do modelo template/app qube.*

Laboratório 9 — Instalando software do jeito certo

Objetivo: instalar um programa em um template e comprovar a herança nos app qubes.

Passo 1 — Inicie o template (por exemplo, **fedora-XX**) abrindo um terminal nele pelo App Menu.

Passo 2 — Instale um programa simples, por exemplo o editor **gedit** ou o utilitário **htop**:

```
sudo dnf install htop      # (ou: sudo apt install htop, em template Debian)
```

Passo 3 — **Desligue o template** e **reinicie** um app qube baseado nele (ex.: **personal**):

```
qvm-shutdown --wait fedora-XX     # no dom0
qvm-shutdown --wait personal && qvm-start personal
```

Passo 4 — Abra um terminal no **personal** e confirme que o programa instalado no template está disponível:

```
which htop && htop --version
```

Passo 5 — Abra a ferramenta **Qubes Update** (ícone na bandeja ou App Menu) e observe a lista de qubes com atualizações pendentes, a coluna "last checked" e o status. **Não** rode atualizações agora se estiver apenas explorando; apenas observe a interface.

Passo 6 — Reflita e anote: por que instalar **htop** no app qube **personal** diretamente (em vez de no template) seria inútil? (Reveja o modelo de persistência do Módulo 7.)

> *Atenção: instalar pacotes com **dnf install/apt install** no template é correto; **não** atualize com **dnf update/apt update** diretos — use a Qubes Update.*

Referências do capítulo

- Qubes OS — How to install software (repositórios padrão; iniciar/instalar/desligar/reiniciar). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-install-software.html
- Qubes OS — Minimal templates (pacotes mínimos; menor superfície de ataque; uso avançado). https://doc.qubes-os.org/en/latest/user/templates/minimal-templates.html
- Qubes OS — Templates. https://doc.qubes-os.org/en/latest/user/templates/templates.html
- Qubes OS — How to update (Qubes Update; QSB; não usar dnf/apt update direto; reiniciar dom0; via Tor). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-update.html
- Qubes OS — Standalones and HVMs. https://doc.qubes-os.org/en/latest/user/advanced-topics/standalones-and-hvms.html

Módulo 10 — Rede e firewall

No Qubes, a rede não é um serviço do sistema — é uma cadeia de qubes. O caminho padrão de um pacote, do mundo físico até o seu navegador, atravessa três compartimentos: o dispositivo físico (placa Wi-Fi/Ethernet) é segurado pelo **sys-net**; o **sys-firewall** fica logo acima, aplicando as regras; e só então o tráfego chega ao seu **app qube**. Cada elo é um qube separado, e essa separação é o que torna a rede — a parte mais exposta de qualquer sistema — incapaz de comprometer o todo.

A documentação recomenda explicitamente essa estrutura em camadas: um qube normalmente não deve se conectar direto ao net qube que tem os dispositivos de rede (o sys-net, por padrão), mas sim a outro net qube primeiro (o sys-firewall, por padrão). O **sys-net**, que toca o hardware e o mundo externo, é o mais exposto; colocar o **sys-firewall** entre ele e seus qubes adiciona uma camada de isolamento e de controle.

O caminho da rede: tudo passa pelos qubes de sistema

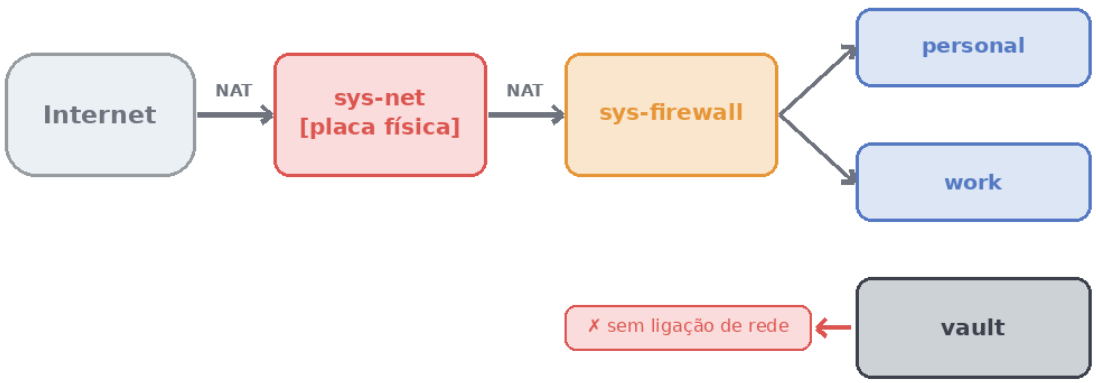


Figura: a rede flui da internet para o sys-net, depois o sys-firewall e os app qubes; o vault fica à parte, sem ligação de rede.

Roteamento, não pontes: eliminando ataques de camada 2

Uma decisão de arquitetura define o tom de segurança da rede. Em vez de "ligar" os qubes em uma ponte (bridge) compartilhada, o Qubes usa roteamento. A documentação explica o porquê: para eliminar ataques de camada 2 vindos de uma VM comprometida, usa-se rede roteada em vez do bridging padrão das interfaces vif, e aplica-se NAT a cada salto.

Traduzindo: se os qubes compartilhassem uma rede em ponte, um qube comprometido poderia montar ataques de camada 2 (ARP spoofing, por exemplo) contra os vizinhos. Com roteamento e NAT a cada salto, isso não é possível — cada qube fala apenas com o roteador acima dele (seu net qube), nunca diretamente com os irmãos. O isolamento de rede não é um efeito colateral; é um objetivo de projeto.

Essas decisões ecoam conceitos clássicos de redes. Em *Redes de Computadores*, Tanenbaum organiza a comunicação em **camadas**: na de **enlace** (camada 2) vivem o Ethernet e o ARP — e ataques como o ARP spoofing; na de **rede** (camada 3) vivem o IP e o **roteamento**.

Escolher **rotear** em vez de **comutar/pontear** os qubes é, literalmente, subir da camada 2 para a 3, o que elimina a classe de ataques de camada 2 entre vizinhos. O **NAT** a cada salto, as tabelas de rota e a tradução de DNS são todos temas centrais desse livro, e revê-los ajuda a entender por que cada qube enxerga apenas o seu gateway, e nunca os irmãos.

Endereçamento: IP, máscara e gateway

Este é o coração técnico do módulo. Entender como o Qubes endereça seus qubes revela, de forma concreta, por que eles não se enxergam.

Como o IP chega ao qube. Não há DHCP. Segundo a doc, o endereço IP da interface eth0 do app qube, além de dois endereços para servir de nameservers (DNS1 e DNS2), são entregues via QubesDB ao app qube durante o seu boot — por isso não é preciso um daemon DHCP no domínio de rede. O IP, a máscara e o gateway são entregues pelo **QubesDB** — o banco de configuração do Qubes — no momento do boot do qube.

A faixa de endereços. Os qubes recebem IPs em uma faixa privada interna, tipicamente **10.137.0.x**. O **sys-net**, por sua vez, pega um IP do mundo real (por exemplo, **192.168.x.x**) via DHCP da rede física.

A tabela de rotas do app qube é mínima. Dentro de um app qube, há essencialmente uma única rota: destino **0.0.0.0**, máscara **0.0.0.0**, saída pela interface **eth0**. Ou seja: "tudo o que não for local, mande para o gateway acima". O qube não conhece rotas para seus vizinhos — ele só sabe falar com o roteador.

A máscara /32 **ponto a ponto.** No net qube (o roteador), a tabela de rotas tem uma entrada por qube conectado, e é aqui que mora a elegância. Cada rota usa a máscara **255.255.255.255** — uma máscara **/32**, que descreve um **único host**. Exemplos reais da documentação:

```
Destino        Gateway   Genmask           Iface
10.137.0.16    0.0.0.0   255.255.255.255   vif4.0
10.137.0.7     0.0.0.0   255.255.255.255   vif10.0
10.137.0.9     0.0.0.0   255.255.255.255   vif9.0
```

Cada qube está ligado ao roteador por um enlace **ponto a ponto** (a interface **vifN.0** corresponde a um qube específico). Uma máscara **/32** significa que aquela "rede" tem exatamente um endereço: o do próprio qube. Não há uma sub-rede compartilhada onde os qubes coabitam. Por isso um qube não tem como endereçar outro diretamente — do ponto de vista do roteamento, cada qube vive sozinho em sua própria rede de um único host, e todo tráfego entre eles teria de passar (e ser autorizado) pelo net qube.

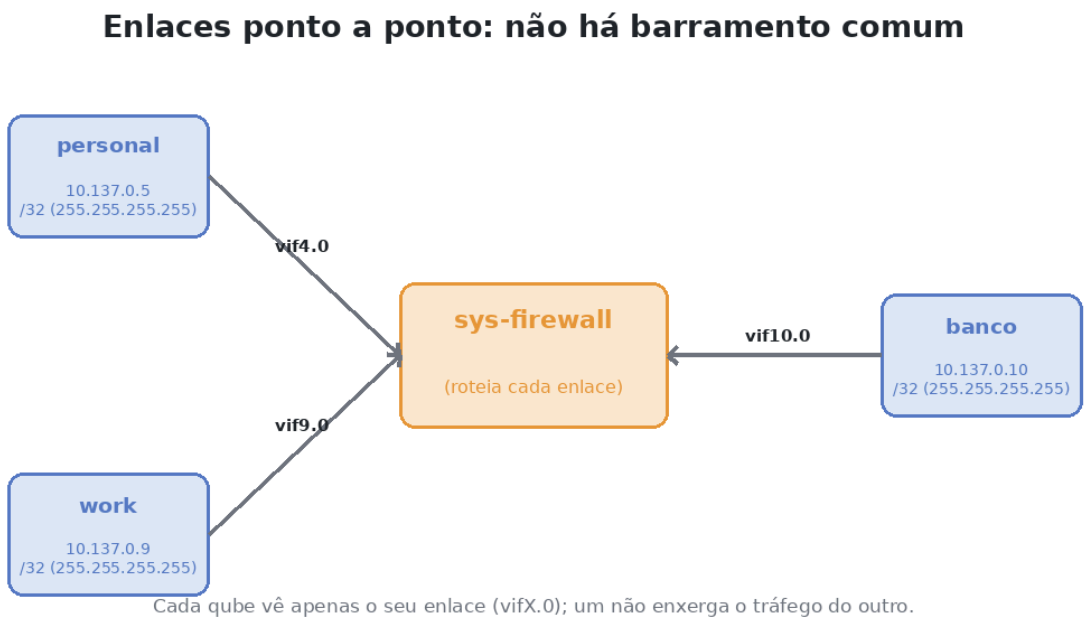


Figura: o sys-firewall liga-se a cada qube por um enlace ponto a ponto próprio (vifX.0), com IP /32 — não há barramento comum entre os qubes.

DNS sem servidor: o truque do DNAT

O DNS também é tratado de forma a não abrir brechas. Os dois nameservers entregues ao qube (**DNS1** e **DNS2**) são, nas palavras da doc, endereços privados — endereços internos, não os DNS reais. Quando uma interface sobe no net qube, o script **qubes_setup_dnat_to_ns** cria as regras DNAT no iptables, traduzindo DNS1 e DNS2 para os servidores DNS reais recém-descobertos.

A vantagem é dupla. Primeiro, a configuração do qube nunca precisa mudar quando você troca de rede (de uma Wi-Fi para outra): o qube sempre aponta para os mesmos **DNS1/DNS2** internos, e o net qube redireciona para os DNS reais do momento. Segundo, e crucial para a segurança: no domínio de rede também não há servidor DNS e, por consequência, não há portas abertas para os qubes. O net qube não roda um servidor DNS exposto aos qubes — ele apenas redireciona pacotes. Menos serviços expostos, menos superfície de ataque.

O firewall: aplicado acima, não dentro

Aqui está o argumento de segurança mais forte sobre a rede, e ele decorre da topologia. No Qubes, o firewall de um qube **não** roda dentro do próprio qube — roda no net qube acima dele. A documentação afirma que todo qube no Qubes OS se conecta à rede por meio de um net qube, que é usado para impor políticas de nível de rede. E, de forma explícita: as regras são implementadas no net qube — se uma regra é definida para um qube que usa o **sys-firewall** como net qube, é o **sys-firewall** que a aplica, pelo serviço **qubes-firewall**.

Por que isso importa tanto? Porque o app qube é, por premissa, não confiável. Se o firewall rodasse *dentro* dele, um malware com privilégios poderia simplesmente desligá-lo. Ao aplicar as regras em um qube *separado* (o **sys-firewall**), o Qubes garante que o compartimento comprometido não tem poder sobre as próprias regras — elas vivem em

outro domínio de confiança. É a diferença entre um preso guardando a própria cela e um guarda do lado de fora.

Detalhes práticos. As regras podem ser definidas pela aba Firewall nas Settings do qube (opções básicas) ou, com mais controle, pelo programa **qvm-firewall** no dom0; elas ficam em **/var/lib/qubes/appvms/<qube>/firewall.xml**. O serviço que as aplica é o **qubes-firewall**, rodando no net qube. E há um comportamento de segurança importante: se esse serviço estiver parado e um novo qube iniciar, o firewall falha fechado — nenhum tráfego passa. Na dúvida, o Qubes bloqueia, não libera.

Uma limitação a conhecer: regras por nome DNS são resolvidas no momento em que as regras entram em vigor, não a cada conexão; por isso não funcionam de forma confiável para servidores que mudam de IP ao longo do tempo por balanceamento de carga baseado em DNS.

> *Reforço de privacidade. O firewall aplicado acima é uma ferramenta direta de controle de vazamento. Você pode, por exemplo, restringir um qube para que ele só alcance um único servidor — e mais nada. Imagine um qube dedicado a um aplicativo de mensagens: limitando-o no **sys-firewall** aos endereços daquele serviço, mesmo que o aplicativo (ou um malware nele) tente "telefonar para casa" para um servidor de espionagem, o pacote morre no net qube. Como a regra vive fora do qube, o malware lá dentro não consegue removê-la. Para quem precisa garantir que um compartimento não fale com ninguém além do estritamente necessário, esse é o mecanismo. E, no extremo, lembre do **vault**: um qube com net qube definido como **None** simplesmente não tem rede — o vazamento por rede torna-se impossível por construção.*

Roteando um qube por uma VPN (net qube de VPN)

A mesma lógica de cadeia de net qubes que serve à rede comum e ao Tor (via Whonix) serve também a uma **VPN**: em vez de instalar o cliente de VPN dentro de cada qube, cria-se um **net qube dedicado à VPN**, e cada qube que deva sair por ela simplesmente o usa como net qube. É a compartimentalização aplicada à rede: um único ponto para configurar, auditar e conter.

O procedimento, na prática: crie um app qube (por exemplo, **sys-vpn**) baseado no template **Fedora** — que já traz os componentes de VPN do NetworkManager; no Debian seria preciso instalá-los no template. Nas Settings, defina o **sys-firewall** como net qube, marque, na aba avançada, a opção que permite que ele **forneça rede a outros qubes**, e habilite o serviço **network-manager**. Copie o arquivo de configuração do provedor (por exemplo, um **.ovpn** do protocolo OpenVPN) para dentro do sys-vpn com o **qvm-copy** (ele chega em **~/QubesIncoming**), importe-o no NetworkManager e conecte. Feito isso, aponte o net qube dos qubes desejados para o **sys-vpn**, e o tráfego deles passa a sair cifrado pela VPN.

Aqui entra o ponto que transforma isso de conveniência em segurança — o **kill-switch**. Sozinha, uma VPN tem um risco silencioso: se o túnel cair, o sistema pode voltar a sair pela rede comum sem você perceber, vazando o tráfego que devia estar protegido. A solução usa o firewall do Qubes, aplicado *acima* (visto na seção anterior): nas regras de firewall do **sys-vpn**, limite as conexões de saída apenas ao **endereço IP do servidor VPN**. Como o firewall roda no net qube acima, nenhum qube que use o sys-vpn consegue alcançar a internet por outro caminho: se a VPN desconecta, todo o resto **morre junto** — nada vaza. É o kill-switch implementado pela topologia, não por um aplicativo que o malware poderia desligar.

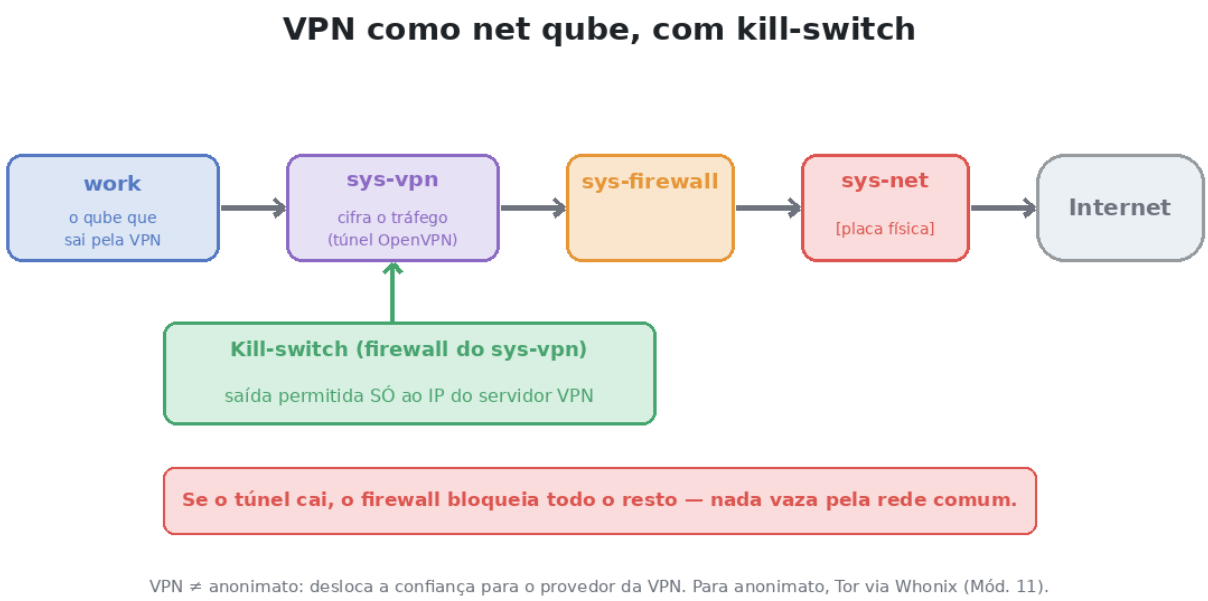


Figura: um qube sai pela VPN através do sys-vpn; o kill-switch — firewall que só libera o IP do servidor VPN — garante que, se o túnel cair, nada vaze pela rede comum.

> *Reforço de privacidade. Uma VPN **não é anonimato**. Ela troca a confiança do seu provedor de acesso (e de quem observa a rede local) pela confiança no **provedor da VPN**, que passa a ver de onde você vem e para onde vai. Para anonimato de verdade, o caminho continua sendo o Tor via Whonix (Módulo 11), onde nenhum nó vê ao mesmo tempo origem e destino. A VPN tem outros usos legítimos — contornar um bloqueio, não confiar no Wi-Fi de um café, separar o tráfego de um compartimento —, e o modelo de net qube dedicado com kill-switch a torna sólida e contida. Use-a por decisão de modelo de ameaças, não por hábito: um túnel a mais não é, por si, mais privacidade.*

Um caso prático vizinho ilustra a fronteira entre USB e rede: usar a internet do celular por cabo (**tethering USB**). Por padrão não funciona — e por um bom motivo: o **sys-usb** (portas USB) e o **sys-net** (rede) são qubes separados, então o "modem USB" do telefone cai no sys-usb, que não faz rede. Dá para contornar movendo o controlador USB correspondente do sys-usb para o sys-net (nas Settings, com os qubes desligados), mas isso expõe aquelas portas USB à rede — reabrindo, em parte, o vetor que o sys-usb existe para fechar (Módulo 3C). A alternativa segura e sem reconfiguração é usar o **hotspot Wi-Fi** do celular, que o sys-net trata como qualquer rede sem fio. Só derrube a separação USB/rede se tiver mesmo necessidade, e devolva-a depois.

Quando o net qube reinicia

Como a rede é uma cadeia de qubes, derrubar um elo afeta os de cima. O Qubes tenta evitar desligar um net qube se há outros qubes em execução que dependem dele. Mas se o net qube parar (um crash, um problema de driver Wi-Fi ao retornar do sleep), a conexão pode precisar de reparo. Em geral o Qubes reconecta sozinho; se não, há um comando no dom0 para restaurar o vínculo, normalmente bastando aplicá-lo ao **sys-firewall**. É um troubleshooting comum, retomado no Módulo 19.

Laboratório 10 — Inspecionando a rede e provando o isolamento

Objetivo: ler o endereçamento real e comprovar que dois app qubes não se enxergam.

Passo 1 — No app qube **personal**, veja IP, máscara e rota:

```
ip addr show eth0
ip route
```

Observe o IP na faixa **10.137.0.x** e a rota padrão (**default via ...**) apontando para o gateway (o net qube). Anote o IP do **personal**.

Passo 2 — Veja os valores entregues pelo QubesDB:

```
qubesdb-read /qubes-ip
qubesdb-read /qubes-netmask
qubesdb-read /qubes-gateway
qubesdb-read /qubes-primary-dns
```

Confirme a máscara **255.255.255.255** (o enlace ponto a ponto).

Passo 3 — Descubra o IP do **work** (repita o Passo 1 nele). Agora, a partir do **personal**, tente alcançar o **work** diretamente:

```
ping -c 2 <IP_do_work>
```

A tentativa deve **falhar** (sem resposta): os qubes não se enxergam, exatamente como a topologia **/32** prevê. Eles só falam com o gateway acima.

Passo 4 — No dom0, inspecione as regras de firewall de um qube:

```
qvm-firewall personal list
```

Passo 5 — Reflita e anote: por que o fato de o firewall rodar no **sys-firewall**, e não no **personal**, garante que um malware no **personal** não consiga abrir conexões proibidas? E como você restringiria um qube para só acessar um único servidor?

Passo 6 — (Opcional, avançado) **VPN com kill-switch.** Monte um **sys-vpn** (app qube Fedora, com o serviço network-manager e "provides network to other qubes"), importe seu **.ovpn** e conecte; aponte o net qube de um qube de teste para o sys-vpn e confirme, com um site de "qual é o meu IP", que a saída mudou. Em seguida, nas regras de firewall do sys-vpn, limite a saída ao IP do servidor VPN e **derrube a VPN**: observe que o qube perde toda a conexão em vez de vazar pela rede comum. (Criação e Settings no dom0; **.ovpn** e NetworkManager dentro do sys-vpn.)

> *Atenção: **qvm-firewall** e comandos **qvm-** rodam no dom0; ip, ping e qubesdb-read*, dentro do qube.*

Referências do capítulo

- Qubes OS — Firewall (enforcement no net qube; qubes-firewall; fail closed; qvm-firewall; firewall.xml; limitação de DNS). https://doc.qubes-os.org/en/latest/user/security-in-qubes/firewall.html
- Qubes OS — Networking (QubesDB; faixa 10.137.0.x; tabelas de rota; máscara 255.255.255.255). https://doc.qubes-os.org/en/latest/developer/system/networking.html
- Qubes OS — VPN (net qube de VPN; NetworkManager; kill-switch por firewall). https://doc.qubes-os.org/en/latest/user/security-in-qubes/vpn.html
- Andrew S. Tanenbaum e David J. Wetherall — *Redes de Computadores* (modelo em camadas; enlace × rede; ARP e ataques de camada 2; roteamento, NAT e DNS;

complementar, fora da doc Qubes).

Módulo 11 — Anonimato com Whonix e Tor

Este é, possivelmente, o módulo mais importante para o público deste livro — e começa com uma distinção que o curso vem repetindo de propósito. **Isolar não é anonimizar.** O Qubes, por padrão, mantém suas atividades separadas umas das outras (segurança por compartimentalização), mas isso não esconde *quem* você é nem *de onde* você se conecta. Um qube comum, mesmo descartável, sai para a internet com o seu endereço IP real.

A documentação é explícita e direta sobre isso: o Qubes OS não promete oferecer propriedades especiais de privacidade (em oposição a segurança) em qubes que não sejam Whonix — e isso inclui os descartáveis. Ou seja: nem mesmo um qube descartável anonimiza você. Para anonimato, o Qubes oferece uma ferramenta específica e integrada: o **Whonix**, que, nas palavras da introdução, roda o Tor com segurança em todo o sistema. A própria política de privacidade do projeto destaca que, desde o momento em que você instala o Qubes OS, ele oferece configurar o Whonix para que todas as suas atualizações passem pelo Tor.

> *Reforço de privacidade. Internalize esta frase antes de prosseguir: se entre seus adversários há quem possa observar de onde você se conecta ou com quem você fala — um provedor, uma operadora, uma autoridade —, então isolamento não basta; você precisa de **anonimato de rede**, e isso significa Tor via Whonix. Confundir os dois é um erro perigoso: muita gente se sente "anônima" por usar um sistema seguro, quando na verdade está apenas usando um sistema bem compartimentado, ainda perfeitamente identificável na rede.*

O que é o Whonix: Gateway e Workstation

O Whonix é um sistema operacional focado em anonimato, integrado ao Qubes como um par de qubes. Quando você marca a opção na instalação (Módulo 5), o Qubes cria dois qubes: o **sys-whonix** (o *Gateway*) e o **anon-whonix** (a *Workstation*). Entender a divisão entre eles é entender por que o Whonix é robusto.

- O **sys-whonix (Gateway)** é um net qube especial: todo o tráfego que passa por ele é forçado a entrar na rede Tor. Ele fica entre a sua workstation anônima e o **sys-net**.
- O **anon-whonix (Workstation)** é onde você efetivamente trabalha (navega com o Tor Browser, por exemplo). Ele usa o **sys-whonix** como net qube, então só consegue alcançar a internet através do Tor.

> *Aprofundamento (fonte: Whonix Project, fora da doc Qubes). A genialidade da separação Gateway/Workstation é o que acontece quando algo dá errado. A Workstation **não tem como saber o seu IP real** — ela não fala diretamente com o hardware de rede, apenas com o Gateway, e o Gateway só lhe oferece Tor. Mesmo que um exploit comprometa completamente a Workstation, o atacante não encontra ali o seu endereço real para vazar, porque ele simplesmente não está disponível naquele qube. Compare com uma configuração de Tor em um sistema comum, onde um navegador comprometido pode, por vários caminhos, descobrir e revelar o IP real da máquina. A arquitetura de dois qubes transforma o anonimato em uma propriedade estrutural, não em uma configuração frágil de aplicativo.*

Tor forçado no gateway: anti-vazamento por design

A propriedade central do Whonix é que o Tor é imposto no **Gateway**, não confiado ao aplicativo. Em um sistema comum, depende do navegador (ou do app) "lembrar" de usar o Tor — e qualquer programa que escape dessa configuração vaza o IP real. No Whonix, a regra é física: como a única saída de rede da Workstation é o Gateway, e o Gateway só fala Tor, **não existe caminho** para um pacote sair sem passar pelo Tor. Um aplicativo mal configurado, ou até malicioso, na Workstation, não consegue "furar" para a rede aberta — não há rota para isso.

Vale conectar com o Módulo 10: o Whonix-Gateway é um caso especial de net qube. A documentação do firewall observa, inclusive, que o Whonix-Gateway (por exemplo, o sys-whonix) não respeita o serviço qubes-firewall — a filtragem dele segue a lógica do Tor, descrita na documentação do Whonix.

> *Reforço de privacidade. "Anti-vazamento por design" é exatamente o que separa uma proteção real de uma ilusão de proteção. Para uma fonte jornalística ou um ativista, configurar manualmente o Tor e torcer para que nenhum aplicativo vaze é inaceitável — um único deslize expõe a identidade. Com o Whonix, o anonimato não depende da disciplina de cada programa: ele é garantido pela arquitetura. Mesmo que você (ou um malware) faça tudo errado dentro da Workstation, o IP real não vaza, porque ele não está ao alcance daquele qube.*

O anonimato de rede também tem raiz na literatura clássica. No capítulo de segurança de redes de *Redes de Computadores*, Tanenbaum descreve a **comunicação anônima** e o **roteamento em camadas (onion routing)**: a mensagem é cifrada em camadas sucessivas e atravessa uma cadeia de relés, cada um conhecendo apenas o anterior e o seguinte — nenhum nó vê, ao mesmo tempo, a origem e o destino. É o princípio do **Tor**, que o Whonix impõe no gateway. Entender o onion routing deixa claro por que o anonimato não vem de "esconder o conteúdo" (isso é cifragem), e sim de **quebrar a ligação entre quem fala e com quem fala** — e por que um único relé comprometido, sozinho, não desfaz a proteção.

Isolamento de fluxos e guardas persistentes

Dois refinamentos do Whonix merecem nota, ambos com impacto direto em anonimato.

> *Aprofundamento (fonte: Whonix Project, fora da doc Qubes). O **isolamento de fluxos (stream isolation)** faz com que aplicativos diferentes usem circuitos Tor diferentes. Sem isso, atividades distintas poderiam sair pelo mesmo circuito e ser correlacionadas — ligando, por exemplo, sua sessão de e-mail anônimo à sua navegação. Com o isolamento, cada aplicativo recebe um caminho próprio na rede Tor, dificultando que um observador junte as peças.*

Já encontramos o segundo refinamento no Módulo 7: os **guardas de entrada persistentes (Tor entry guards)**. Lembre que o **sys-whonix** usa **bind-dirs** para tornar persistente o diretório **/var/lib/tor**, preservando seus guardas entre reinicializações. Os guardas são os primeiros relés por onde seu tráfego entra no Tor; mantê-los estáveis é uma defesa contra um adversário que tente, ao longo do tempo, fazer você rotacionar por relés que ele controla para desanonimizá-lo.

Atualizando por Tor

Como visto nos Módulos 5 e 9, o Whonix permite rotear as atualizações do sistema por Tor. A política de privacidade do projeto trata isso como recurso de primeira classe — já na

instalação, o Qubes oferece configurar o Whonix para que todas as atualizações passem pelo Tor. Isso impede que um observador de rede saiba quando você atualiza, quais pacotes baixa e, por correlação, qual versão do sistema você usa. A única ressalva prática, já mencionada, é que atualizar por Tor pode demorar bastante.

O que o Whonix não resolve

Fiel ao método do curso, é preciso dizer onde o anonimato encontra seus limites. A documentação não esconde: privacidade é muito mais difícil do que se costuma imaginar. Além do navegador, há também o fingerprinting de VM e ataques avançados de desanonimização que a maioria dos usuários nunca considerou. O Whonix se especializa em proteger contra muitos desses riscos, mas a doc fecha com uma frase que você deve levar a sério: o Whonix é uma ferramenta poderosa, mas nenhuma ferramenta é perfeita — leia a documentação com atenção e use-o com cuidado.

Em termos práticos, isto significa: o Tor esconde seu IP, mas não protege contra o que *você* revela (fazer login em uma conta identificável dentro do anon-whonix entrega sua identidade, por mais anônima que seja a rede). Não protege contra erros de comportamento, contra correlação de tráfego por um adversário global, nem contra você baixar e abrir, fora do Whonix, algo que telefone para casa. Anonimato é uma prática contínua, não um botão.

> *Reforço de privacidade. A lição operacional para o público deste livro é dividir identidades por qubes e jamais misturá-las. Use o **anon-whonix** apenas para a atividade anônima, e nunca acesse nele algo ligado à sua identidade real — nem por um instante, nem "só para checar". A correlação entre uma atividade anônima e uma identificável, feita na mesma sessão ou pelo mesmo padrão de uso, é uma das formas mais comuns de desanonimização, e nenhuma ferramenta a impede por você. O Whonix entrega o canal anônimo; a disciplina de não contaminá-lo é sua.*

Laboratório 11 — Navegação anônima e teste de vazamento

Objetivo: usar o Whonix e comprovar que o tráfego sai pelo Tor, distinguindo na prática segurança de anonimato.

Passo 1 — Confirme a topologia. No dom0:

```
qvm-prefs anon-whonix netvm
```

O resultado deve ser **sys-whonix** — a Workstation usa o Gateway como net qube.

Passo 2 — No **anon-whonix**, abra o Tor Browser (pelo App Menu) e acesse **https://check.torproject.org**. Confirme a mensagem de que você está usando o Tor.

Passo 3 — **Compare.** Abra um navegador em um qube comum (ex.: **personal**) e acesse o mesmo endereço ou um site que mostre seu IP. Observe que ali aparece seu IP/região reais — prova concreta de que o qube comum **não** anonimiza.

Passo 4 — Anote os dois IPs/resultados lado a lado: o do **anon-whonix** (um nó de saída Tor, em outro país) e o do **personal** (seu IP real). Essa diferença é a distinção entre segurança e anonimato, tornada visível.

Passo 5 — Reflita e anote: se você fizesse login em uma conta pessoal identificável dentro do **anon-whonix**, o Tor o protegeria? Por quê? (Reveja o reforço sobre não misturar identidades.)

*Atenção: não acesse, no **anon-whonix**, nada ligado à sua identidade real. Trate-o como um compartimento exclusivamente anônimo.*

Referências do capítulo

- Qubes OS — FAQ (privacidade é mais difícil do que parece; fingerprinting de VM; nenhuma ferramenta é perfeita). https://doc.qubes-os.org/en/latest/introduction/faq.html
- Qubes OS — Privacy policy (Whonix oferecido na instalação; updates via Tor). https://doc.qubes-os.org/en/latest/introduction/privacy.html
- Qubes OS — Installation guide (criação de sys-whonix e anon-whonix). https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/installation-guide.html
- Whonix — Qubes-Whonix (complementar, fora da doc Qubes). https://www.whonix.org/wiki/Qubes
- Qubes OS — Firewall (nota sobre Whonix-Gateway). https://doc.qubes-os.org/en/latest/user/security-in-qubes/firewall.html
- Qubes OS — How to make any file persistent (bind-dirs; Tor entry guards no sys-whonix). https://doc.qubes-os.org/en/latest/user/advanced-topics/bind-dirs.html
- Whonix — Stream Isolation (complementar). https://www.whonix.org/wiki/Stream_Isolation
- Qubes OS — How to update (atualizações via sys-whonix). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-update.html
- Whonix — Documentation (complementar). https://www.whonix.org/wiki/Documentation
- Andrew S. Tanenbaum e David J. Wetherall — *Redes de Computadores* (segurança de redes; comunicação anônima e onion routing; complementar, fora da doc Qubes).

Módulo 12 — Dispositivos: USB, armazenamento, câmera e microfone

Em um sistema comum, conectar um pendrive, uma câmera ou um microfone os disponibiliza imediatamente para todo o sistema. No Qubes, nada é conectado a um qube sem a sua decisão explícita — o mesmo princípio de mediação que rege o clipboard e a cópia de arquivos vale para o hardware. A documentação descreve quatro classes que o Qubes entende: **microfones**, **dispositivos de bloco**, **dispositivos USB** e **dispositivos PCI**.

A interface para lidar com eles foi unificada na versão 4.0 em torno do comando **qvm-device** e do Qubes Devices Widget — o ícone de quadrado amarelo na bandeja (Módulo 6). Microfones, dispositivos de bloco e dispositivos USB podem ser anexados pelo widget; dispositivos PCI, pelas Settings do qube, mas exigem reiniciar o qube.

> *Reforço de privacidade. O princípio de "anexar sob demanda" é, em si, uma proteção de privacidade. Um microfone ou uma câmera que não estão anexados a nenhum qube simplesmente não existem para o software — não há como um aplicativo, mesmo malicioso, gravar você se o dispositivo não lhe foi entregue. Para quem teme vigilância, isso é concreto: em vez de confiar que nenhum programa vai ligar sua câmera, você garante que a câmera não está disponível para programa nenhum até você, deliberadamente, anexá-la a um qube específico — e a desanexa assim que termina.*

Anexar e desanexar na prática

Pelo **widget**: clique no ícone, passe o mouse sobre o dispositivo desejado e escolha, na lista de qubes em execução, aquele a que quer anexá-lo. Para desanexar, clique de novo no ícone — dispositivos anexados aparecem em **negrito** — e use o símbolo de ejetar ao lado do qube.

Pela **linha de comando** do dom0, há ferramentas por classe. Para USB, o **qvm-usb**:

```
qvm-usb                              # lista os dispositivos USB disponíveis
qvm-usb attach work sys-usb:2-5      # anexa o dispositivo 2-5 (do sys-usb) ao qube 'work'
qvm-usb detach work sys-usb:2-5      # desanexa
```

A saída de **qvm-usb** identifica cada dispositivo de forma legível — por exemplo, **sys-usb:2-5 058f:3822 058f_USB_2.0_Camera**. Note a lógica: o dispositivo físico vive no **sys-usb**, e você o "empresta" temporariamente a um qube de trabalho. Há um detalhe sobre compartilhamento: apenas o microfone pode ser anexado a mais de um qube em execução ao mesmo tempo; os demais, a um por vez.

sys-usb e o perigo do USB

Retomamos aqui, com profundidade prática, o Mecanismo 2 do Módulo 3C. Anexar USB a um qube exige um **sys-usb** — a documentação é direta: usar dispositivos USB em qubes requer um qube USB. E ela não economiza no alerta: o passthrough de USB traz muitas implicações de segurança.

A razão está em como o USB funciona. Toda a pilha USB precisa interpretar os dados que o dispositivo apresenta para descobrir o que ele é — e um dispositivo malicioso pode explorar bugs nesse processo ou, pior, simplesmente se passar por um teclado. É o ataque BadUSB,

com implementações prontas como o USB Rubber Ducky: um "pendrive" que, na verdade, é um teclado que digita comandos sozinho. Daí o alerta mais grave da doc sobre dispositivos de entrada: se você conecta dispositivos USB de entrada (teclado e mouse) a um qube, esse qube passa a ter, na prática, controle sobre o seu sistema — e ainda pode farejar tudo que você digita, como senhas.

Por isso o **sys-usb** existe: ele confina o controlador USB inteiro em um qube isolado (de preferência descartável). Um BadUSB conectado ataca, no máximo, o **sys-usb**, e não o dom0 nem seus qubes de trabalho.

> *Reforço de privacidade. O cenário do "dispositivo emprestado" é um vetor clássico contra jornalistas e ativistas: um carregador, um pendrive "com um documento", um cabo de presente. Em um sistema comum, espetar isso pode comprometer tudo. No Qubes, o **sys-usb** recebe o golpe e o contém; e como você decide a qual qube (se algum) o dispositivo será anexado, um "pendrive" que se finge de teclado não ganha controle do seu sistema — ele fica preso, inútil, no compartimento USB. A disciplina de nunca anexar um dispositivo desconhecido a um qube sensível é parte da sua higiene de privacidade.*

Prefira o mínimo: dispositivo de bloco em vez de USB inteiro

Há uma regra de ouro para dispositivos, e ela é a aplicação direta do princípio do menor privilégio: toda capacidade extra que um qube ganha é superfície de ataque adicional; por isso, convém sempre anexar a um qube apenas o mínimo necessário.

Na prática, para um pendrive de armazenamento, prefira anexá-lo como **dispositivo de bloco** em vez de USB inteiro — a documentação recomenda anexar, sempre que possível, um dispositivo de bloco em vez do USB inteiro. Por quê? Anexar o dispositivo USB completo expõe o qube à pilha USB inteira; anexar apenas o dispositivo de bloco (ou, melhor ainda, uma única partição) reduz o que o qube precisa processar. Como diz a doc de segurança, anexar uma partição única evita que o qube tenha de interpretar a tabela de partições. Menos a interpretar, menos a explorar.

Dispositivos PCI

Os dispositivos PCI (placas de rede, controladores USB) são o caso mais sensível, e por isso o Qubes os trata de forma especial — é assim que o **sys-net** e o **sys-usb** "seguram" seu hardware. A documentação adverte que anexar um dispositivo PCI a um qube tem sérias implicações de segurança, porque o qube ganha controle direto sobre o hardware e fica exposto a bugs ou a implementações maliciosas dele. Por padrão, o Qubes exige que o dispositivo seja resetável pelo hypervisor (Módulo 3C), de modo que possa ser reinicializado e voltar a ser confiável. Anexar ou remover um PCI exige reiniciar o qube. Para a maioria dos usuários, os dispositivos PCI já estão atribuídos aos qubes de sistema na instalação e não precisam ser mexidos.

Microfone e câmera sob demanda

O microfone é uma classe própria no Qubes, e a câmera, na maioria dos laptops, é um dispositivo USB — a doc observa que, na maioria dos laptops, hardware integrado como câmeras e leitores de impressão digital é implementado como dispositivos USB. O ponto operacional é o mesmo: eles só ficam disponíveis para um qube quando você os anexa, e você deve desanexá-los assim que terminar.

Reforço de privacidade. Esta é, talvez, a proteção de privacidade mais palpável do Qubes no dia a dia. Em um sistema comum, malware ou um aplicativo abusivo pode ligar seu microfone ou sua câmera sem que você perceba — espionagem ambiente é uma ameaça real para quem está sob vigilância. No Qubes, o microfone só grava em um qube ao qual você o anexou explicitamente, e a câmera idem. Faça disso um hábito: anexe o microfone ao qube de videochamada apenas durante a chamada, e o desanexe ao desligar. Fora desses momentos, nenhum software no sistema tem como ouvir você — porque o dispositivo simplesmente não lhe foi dado.

Laboratório 12 — Anexando dispositivos com segurança

Objetivo: praticar o anexar/desanexar e sentir o controle explícito sobre o hardware.

Passo 1 — No dom0, liste os dispositivos USB disponíveis e os de bloco:

```
qvm-usb
qvm-block
```

Passo 2 — Espete um pendrive. Pelo **Qubes Devices Widget**, anexe-o como **dispositivo de bloco** (não USB inteiro) ao qube **personal**. No **personal**, confirme que ele apareceu (ex.: **lsblk**) e monte-o se necessário.

Passo 3 — Quando terminar, **desanexe** o pendrive pelo widget (símbolo de ejetar). Confirme no **personal** que ele sumiu.

Passo 4 — Anexe o **microfone** a um qube específico pelo widget; confirme que ele está disponível ali e, em seguida, desanexe-o.

Passo 5 — Reflita e anote: por que anexar um pendrive como dispositivo de bloco é mais seguro do que como USB inteiro? E por que deixar o microfone desanexado por padrão protege sua privacidade melhor do que confiar que nenhum app vai ligá-lo?

Passo 6 — **Câmera e microfone de verdade.** No template (ex.: fedora-XX ou debian-XX), instale um app de teste — por exemplo, o **cheese** (câmera) — e, no menu de aplicativos do template nas Settings do qube, adicione-o. **Desligue o template** e reinicie o app qube: sem esse desligamento, o programa recém-instalado não aparece (a herança do Módulo 7). Abra o **cheese** no app qube: ele acusará "nenhuma câmera" — porque a câmera ainda não foi entregue a ele.

Passo 7 — Pelo **Qubes Devices Widget**, anexe o **microfone** e a **câmera integrada** ao app qube. Reabra o **cheese**: agora a câmera funciona. Dispositivos anexados aparecem em **negrito** no widget; ejete-os ao terminar. Atenção prática: **nunca** anexe a um qube o receptor USB do seu mouse/teclado sem fio — o app qube passa a controlar o dispositivo e você perde mouse/teclado na hora. E, se disponível no seu sys-usb, ative a opção de **perguntar antes de conectar** um novo dispositivo: um passo a mais contra o USB malicioso do Módulo 3C.

Atenção: nunca anexe um dispositivo USB desconhecido (recebido, emprestado, "achado") a um qube sensível. Na dúvida, use um qube descartável e prefira o modo bloco.

Referências do capítulo

- Qubes OS — How to use devices (quatro categorias; qvm-device; Devices Widget). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-use-devices.html

- Qubes OS — How to use USB devices (exige USB qube; implicações de segurança). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-use-usb-devices.html
- Qubes OS — Device handling security (BadUSB; Rubber Ducky; dispositivos de entrada). https://doc.qubes-os.org/en/latest/user/security-in-qubes/device-handling-security.html
- Qubes OS — How to use block storage devices. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-use-block-storage-devices.html
- Qubes OS — How to use PCI devices. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-use-pci-devices.html

Módulo 13 — Qubes descartáveis em profundidade

Para que serve um qube descartável

O qube descartável (disposable) é, talvez, a ferramenta mais característica e mais útil do Qubes no dia a dia de quem precisa de segurança. Já apareceu como o Mecanismo 5 do Módulo 3C (não-persistência) e como a resposta correta, no Módulo 7, para o problema de que reiniciar um app qube comum não o limpa de verdade. Agora, em profundidade.

A documentação lista os casos de uso, que vale ler como um cardápio das suas defesas cotidianas:

- visualizar arquivos não confiáveis sem que haja outros arquivos confiáveis no mesmo sistema;
- visitar sites não confiáveis em um navegador sem os cookies de autenticação de sessões anteriores;
- sanitizar PDFs ou imagens não confiáveis e obter de volta uma versão segura de visualizar;
- conectar dispositivos não confiáveis isolados dos dispositivos ou arquivos confiáveis.

Em comum, todos tratam do **não confiável**: o arquivo que você recebeu, o site que você não conhece, o dispositivo que lhe deram. O disposable é o compartimento de uso único onde você lida com isso e que depois deixa de existir.

Como funcionam: do disposable template ao disp1234

A mecânica tem três peças. Primeiro, o **disposable template** — por padrão, o **default-dvm**. Ele não é, em si, um descartável, mas um template especial capaz de criar diferentes tipos de descartável. É o molde a partir do qual os descartáveis nascem.

Segundo, o **disposable** propriamente dito. A documentação descreve seu ciclo de vida com precisão: os descartáveis nascem dos disposable templates, não têm nome fixo e são apagados do sistema assim que se fecha o aplicativo inicial aberto neles — portanto, qualquer mudança feita no descartável se perde. Na prática: você lança um aplicativo a partir do **default-dvm** e um novo qube descartável chamado **disp1234** (onde 1234 é um número aleatório) inicia e abre o aplicativo escolhido; ao fechar a janela do aplicativo, o qube **disp1234** desliga e desaparece do seu sistema.

Esse é o ponto que o diferencia de tudo o que foi visto: o disposable não tem sistema de arquivos de usuário persistente. Lembre do Módulo 7 — em um app qube comum, malware pode se esconder no **/home**, que persiste. No disposable, não há **/home** que sobreviva: ele inicia completamente limpo a cada vez. É a única não-persistência verdadeiramente completa do Qubes.

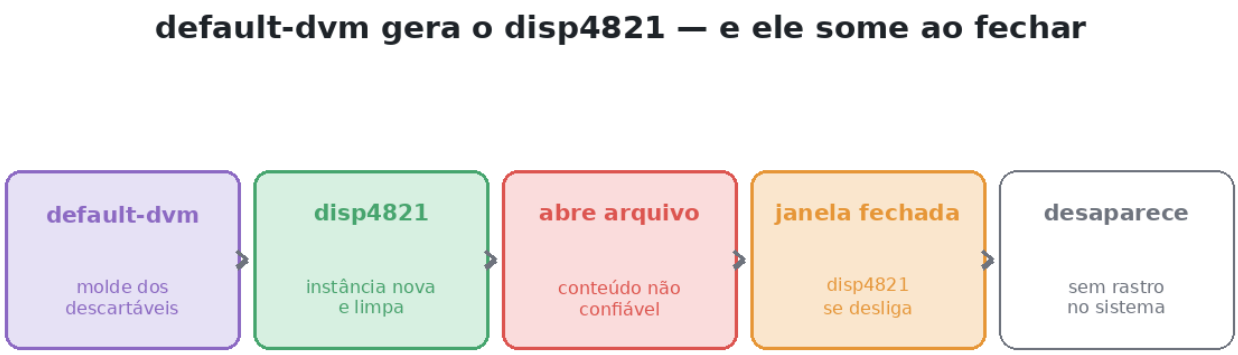


Figura: o default-dvm gera um descartável limpo (disp4821) que abre o arquivo não confiável e, ao fechar a janela, desaparece sem rastro.

Abrindo anexos e links não confiáveis

O uso mais valioso é abrir o desconhecido sem risco. A documentação dá o exemplo perfeito: em um qube de e-mail, o usuário recebe um anexo PDF; usando o protocolo qrexec (**qubes.OpenInVM**), o PDF é aberto em um novo qube descartável. O PDF — que pode ser uma arma — é aberto não no qube de e-mail, onde estão suas mensagens, mas em um descartável que será destruído depois. Mesmo que o arquivo explore um bug no leitor, o estrago morre com o disposable.

Na prática, você faz isso pelo gerenciador de arquivos (botão direito → abrir em disposable) ou abrindo o aplicativo diretamente a partir do **default-dvm** no App Menu (por exemplo, **APPS → default-dvm → Firefox** abre um navegador novo e descartável).

> *Reforço de privacidade. Este é o fluxo que protege uma fonte. Quando você recebe um documento por um canal sensível, abri-lo no mesmo qube onde guarda suas comunicações seria arriscar que um anexo armadilhado comprometa tudo — e, pior, persista para espionar suas próximas interações. Abrindo em um disposable, o documento é examinado em um compartimento que nasce limpo e desaparece por completo ao fechar: nenhum gancho sobrevive, nenhum rastro fica. Para quem lida com material de origem desconhecida como rotina, "abrir em disposable" deve ser o reflexo padrão, não a exceção.*

Sanitizando PDFs: transformando o perigoso em seguro

Um caso especial merece destaque, porque é um dos truques mais elegantes do Qubes: a sanitização de PDFs. Em vez de apenas *abrir* um PDF suspeito em um disposable, você pode convertê-lo em uma versão segura. O processo (o conversor confiável de PDF citado na arquitetura, Módulo 3C) abre o arquivo em um disposable, renderiza cada página como imagem e devolve um novo PDF feito dessas imagens — uma versão segura de visualizar. O resultado não contém mais nenhum código ativo, nenhuma fonte maliciosa, nenhum script: apenas pixels. O original perigoso é processado e descartado dentro do disposable; só a versão inofensiva volta para você.

> *Reforço de privacidade. A conversão de PDF é a materialização da "regra de direção" do Módulo 8 aplicada a documentos: em vez de trazer um arquivo não confiável para um*

> *qube sensível, você traz apenas o seu resíduo seguro. Para um jornalista que precisa ler um dossiê recebido de uma fonte sem saber se ele esconde um exploit, sanitizar o PDF em um disposable permite consultá-lo no qube de trabalho com risco drasticamente reduzido. O conteúdo chega; o perigo fica para trás, no compartimento que some.*

Named disposables: descartáveis com nome

Há uma variante: o **named disposable** (descartável nomeado). Diferente do descartável comum, ele tem nome fixo e, nas palavras da doc, comporta-se como um app qube comum: não desliga quando você fecha o aplicativo inicial, todos os aplicativos que você abrir iniciam no mesmo qube, e é preciso desligá-lo manualmente. A semelhança com um app qube, porém, para na persistência: ao ser desligado, qualquer mudança feita no descartável nomeado se perde.

Você já usa named disposables sem saber: se marcou na instalação a opção de qubes de sistema descartáveis, seu sys-net, sys-usb e sys-firewall são exemplos de descartáveis nomeados baseados no disposable template **default-dvm**. É assim que a borda do sistema (Módulo 7) ganha a não-persistência: esses qubes têm um papel fixo e nome estável, mas nascem limpos a cada inicialização.

Customizando seus descartáveis

Como tornar um programa disponível em todos os descartáveis, ou fixar uma configuração? A resposta segue a lógica de herança do Módulo 7. Se você precisa de mudanças persistentes nos arquivos do descartável (por exemplo, arquivos de configuração), faça-as no disposable template. E para instalar programas: se você precisa que certos programas estejam disponíveis no descartável, instale-os no template em que o próprio disposable template se baseia — ou seja, no template comum subjacente. Você pode ter quantos disposable templates quiser, cada um afinado para um propósito (um para navegação, um para abrir documentos, etc.). Para criar um, basta marcar a opção **Disposable template** nas Settings de um app qube customizado.

Um lembrete importante: descartável não é anônimo

Antes da síntese, uma ressalva que conecta com o Módulo 11. O disposable é limpo, mas **não** é anônimo. A documentação foi explícita ao dizer que o Qubes não promete oferecer propriedades especiais de privacidade em qubes que não sejam Whonix — e isso inclui os descartáveis. Um descartável comum sai para a internet com o seu IP real. Para abrir algo de forma anônima *e* descartável, combine os dois: use um disposable baseado em um disposable template Whonix. Não confunda "sem rastro local" com "sem identificação na rede".

Laboratório 13 — Abrindo o não confiável sem risco

Objetivo: usar disposables para navegação e para abrir um arquivo, observando o ciclo de vida.

Passo 1 — Abra um navegador descartável: App Menu → APPS → **default-dvm** → Firefox. No dom0, confirme que surgiu um qube **dispXXXX**:

```
1 qvm-ls --running | grep disp
```

Passo 2 — Anote o nome (**dispXXXX**). Feche a janela do navegador e confirme, repetindo o comando acima, que o qube **desapareceu**.

Passo 3 — No **personal**, crie um arquivo de teste (ex.: um **.txt** ou um PDF qualquer), clique com o botão direito no gerenciador de arquivos e escolha abri-lo em um disposable. Observe que ele abre em um **dispXXXX**, não no **personal**.

Passo 4 — Se disponível, experimente a opção de **converter/sanitizar** um PDF e compare o resultado com o original.

Passo 5 — Reflita e anote: por que abrir um anexo suspeito em um disposable é mais seguro do que abri-lo no **personal** e depois reiniciar o **personal**? (Reveja o limite da não-persistência no Módulo 7.) E o que você acrescentaria para abri-lo de forma também anônima?

> *Atenção: o disposable protege o resto do sistema, mas o que você faz dentro dele ainda importa — não faça login em contas identificáveis em um disposable comum se o seu objetivo for privacidade.*

Referências do capítulo

- Qubes OS — How to use disposables (disposable template; default-dvm; ciclo dispXXXX). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-use-disposables.html
- Qubes OS — Glossary (disposable). https://doc.qubes-os.org/en/latest/user/reference/glossary.html
- Qubes OS — Architecture (trusted PDF and Image converters). https://doc.qubes-os.org/en/latest/developer/system/architecture.html
- Qubes OS — Disposable customization. https://doc.qubes-os.org/en/latest/user/advanced-topics/disposable-customization.html
- Qubes OS — FAQ (privacidade em qubes não-Whonix, incluindo disposables). https://doc.qubes-os.org/en/latest/introduction/faq.html

Módulo 14 — Chaves e segredos

Suas chaves criptográficas, senhas e frases-semente são o seu bem digital mais valioso e mais frágil. Em um sistema comum, elas vivem no mesmo espaço onde você navega, lê e-mails e abre anexos — exatamente as atividades mais expostas. Basta um comprometimento para que um atacante copie a chave privada e, com ela, decifre tudo o que é seu, para sempre. Este módulo mostra como o Qubes resolve isso isolando os segredos do uso dos segredos.

A peça central é o **vault**: um qube sem rede, criado por padrão na instalação. Como visto no Módulo 3, basta definir o net qube de um qube como **None** para que ele fique totalmente offline. Um segredo guardado nele não pode ser exfiltrado pela rede, porque não há rede — nem que algum software dentro do vault tente. Mas há um problema aparente: se o vault não tem rede, como você usa a chave que está nele para assinar um e-mail no qube de e-mail? A resposta é a família de técnicas *split*.

Split-GPG: usar a chave sem expor a chave

O Split-GPG é a ideia mais elegante do módulo. A documentação o descreve assim: ele implementa um conceito parecido com ter um smart card com suas chaves GPG privadas, exceto que o papel do "smart card" é desempenhado por outro app qube do Qubes. Em vez de um cartão inteligente físico, o "cofre" é outro qube — o vault.

O mecanismo: um domínio não tão confiável — por exemplo, aquele onde o Thunderbird roda — pode delegar todas as operações de criptografia (cifragem, decifragem e assinatura) a outro domínio, mais confiável e isolado da rede. O qube de e-mail nunca vê a chave privada; ele apenas envia ao vault o pedido "assine isto" ou "decifre aquilo", e recebe de volta o resultado. A chave permanece o tempo todo no vault isolado. A consequência é decisiva: comprometer o domínio onde o Thunderbird ou outro app cliente roda não permite ao atacante roubar automaticamente todas as suas chaves.

A documentação faz ainda uma observação afiada sobre por que a abordagem comum (proteger a chave com uma senha) é insuficiente: as tão usadas senhas em chaves privadas são praticamente inúteis, porque o atacante pode facilmente montar um backdoor simples que espera o usuário digitar a senha e então rouba a chave. Uma senha não protege uma chave que está no mesmo lugar comprometido — o atacante só espera você digitá-la. O Split-GPG resolve isso tirando a chave do lugar exposto por completo.

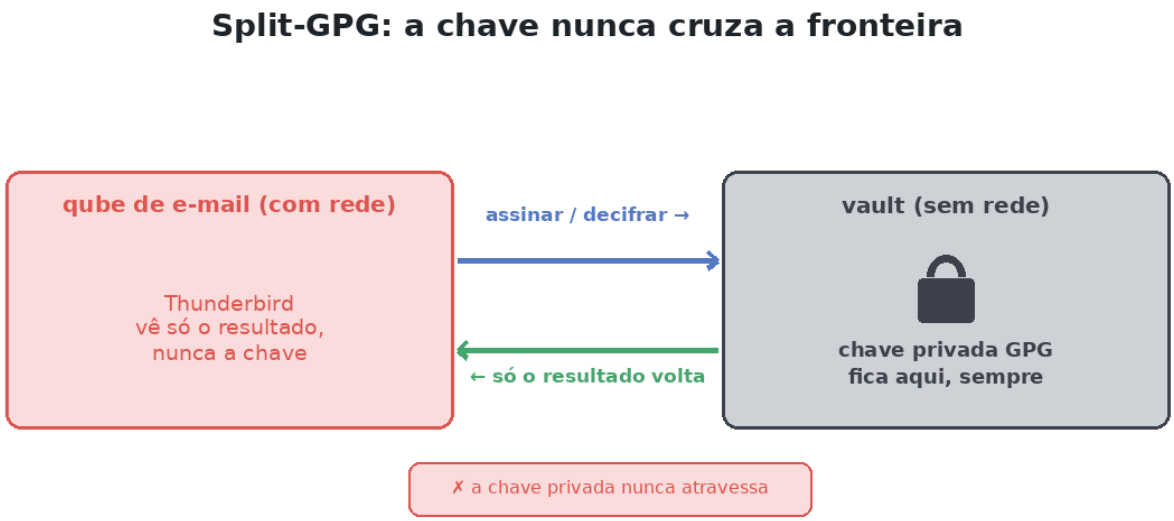


Figura: o qube de e-mail pede ao vault para assinar/decifrar e recebe só o resultado; a chave privada nunca cruza a fronteira do vault.

Melhor que um smart card — e onde está o limite

Costuma-se achar que um smart card físico é o ápice da segurança de chaves. A documentação contesta isso com um argumento importante: o smart card protege a chave em si, mas, em geral, nada impede o atacante de pedir ao cartão que decifre todos os documentos do usuário que ele tenha encontrado. O chip não tem como saber que os pedidos de decifração agora vêm do script do atacante, e não do usuário. Ou seja: a chave não vaza, mas o atacante ainda manda o cartão decifrar tudo.

O Split-GPG ataca também esse flanco: a cada vez que a chave vai ser usada, o usuário é consultado para dar consentimento (com um tempo-limite configurável, de 5 minutos por padrão), e é sempre avisado de cada uso por uma notificação na bandeja. Você vê — e autoriza — cada uso da chave, e recebe uma notificação visível. Um pedido inesperado de decifração salta aos olhos.

Fiel ao método do curso, a doc também aponta onde está o limite: o problema de *confiabilidade da assinatura* não é resolvido. O usuário não tem como saber o que o chip está realmente assinando, e esse problema de confiabilidade da assinatura não é solucionável pelo Split-GPG. Em outras palavras: o Split-GPG garante que a chave não vaza e que cada uso é consentido, mas não garante que o que está sendo assinado é exatamente o que você pensa — um qube cliente comprometido poderia lhe mostrar um texto e enviar outro para assinatura. Saber disso é parte de operar com realismo.

> *Reforço de privacidade. Para quem usa GPG para proteger comunicações com fontes ou contatos, o Split-GPG muda o jogo. Mesmo que o qube onde roda o cliente de e-mail seja totalmente comprometido — cenário que a doc trata como nada improvável —, a chave privada que identifica você e protege anos de correspondência permanece intocada no vault. E a notificação a cada uso transforma um roubo silencioso em um evento visível: se a chave for usada quando você não pediu, você fica sabendo. Privacidade de longo prazo depende de a chave nunca cair; o Split-GPG é o que torna isso possível em um computador conectado.*

Split-SSH: o mesmo princípio para chaves SSH

O mesmo conceito se aplica a chaves SSH. No Split-SSH, sua chave privada SSH fica em um vault isolado, e os qubes que precisam se conectar a servidores delegam a autenticação a esse vault, sem nunca verem a chave. Se o qube de trabalho for comprometido, o atacante não consegue copiar sua chave SSH — no máximo, e apenas enquanto você autoriza, usá-la para uma conexão, que você pode ver e negar. É a guia comunitária de Split-SSH, referenciada pela documentação externa do Qubes.

U2F e o CTAP proxy: 2FA sem expor o navegador

A autenticação de dois fatores por hardware (chaves de segurança U2F/FIDO2) é uma defesa excelente, mas conectá-la diretamente ao qube do navegador exporia esse qube à pilha USB inteira — o risco do Módulo 12. O Qubes resolve com o **CTAP proxy**, descrito na doc como um proxy seguro para usar dispositivos de autenticação de dois fatores CTAP com navegadores, sem expor o navegador à pilha USB inteira. É o mesmo princípio dos proxies de teclado/mouse: a chave física fica no **sys-usb**, e apenas as mensagens de autenticação — não o acesso USB bruto — chegam ao navegador.

A documentação situa os padrões: U2F (CTAP1) é um método robusto de autenticação de segundo fator que usa chaves de segurança de hardware dedicadas; FIDO2 (CTAP2 + WebAuthn) estende isso para uma autenticação robusta e sem senha. O ganho prático: você usa sua chave de segurança normalmente, mas o navegador — o software mais exposto — nunca toca o USB.

> *Reforço de privacidade. Para um alvo de ataques direcionados, a 2FA por hardware é uma das defesas mais eficazes contra phishing e roubo de credenciais — e o CTAP proxy permite usá-la sem abrir o navegador à superfície de ataque do USB. É segurança somada à segurança: a chave física protege suas contas, e o proxy protege o sistema da chave.*

O gerenciador de senhas no vault

A aplicação mais cotidiana de tudo isso é guardar senhas em um gerenciador (como o KeePassXC) dentro do vault, offline. Quando precisar de uma senha, você a copia do vault para o qube de destino usando o clipboard seguro do Módulo 8 (**Ctrl+Shift+C** / **Ctrl+Shift+V**), que entrega o conteúdo apenas ao qube escolhido e zera o clipboard global depois. Assim, o banco de senhas inteiro nunca está exposto a um qube com rede: só a senha do momento viaja, e só para onde você mandou.

> *Reforço de privacidade. Lembre do exemplo do Módulo 8: sem o clipboard seguro, copiar uma senha do vault a vazaria para todos os qubes em execução, inclusive o **sys-net**. Com o vault offline + clipboard mediado, o seu cofre de senhas permanece inalcançável pela rede, e cada senha sai dele apenas sob seu comando explícito e para um único destino. Para quem protege muitas contas sensíveis, essa é a diferença entre um cofre e uma gaveta aberta.*

Laboratório 14 — Segredos isolados na prática

Objetivo: experimentar o isolamento de segredos com o vault e (opcionalmente) o Split-GPG.

Passo 1 — Confirme que o **vault** está offline:

```
qvm-prefs vault netvm        # deve retornar vazio/None (no dom0)
```

Passo 2 — No **vault**, abra um gerenciador de senhas (ex.: KeePassXC, se instalado no template) e crie uma entrada de teste. Copie a senha com **Ctrl+C** e depois **Ctrl+Shift+C**.

Passo 3 — Vá a um app no **personal**, **Ctrl+Shift+V** e **Ctrl+V**. Observe que a senha foi entregue apenas ao **personal** e que o clipboard global foi zerado.

Passo 4 — (Opcional, avançado) Configure o **Split-GPG** seguindo a documentação: gere/importe sua chave no **vault**, instale **qubes-gpg-split-dom0** no dom0 e o cliente no qube de e-mail. Ao assinar/decifrar uma mensagem, observe o **prompt de consentimento** e a **notificação** vindos do vault.

Passo 5 — Reflita e anote: por que proteger a chave privada apenas com uma senha (sem Split-GPG) não basta? (Reveja a frase da doc sobre o backdoor que espera a senha.) E por que a notificação a cada uso aumenta sua segurança?

> *Atenção: nunca dê rede ao **vault**. Se precisar instalar software para o gerenciador de senhas, faça-o no **template** do vault, não no vault em si.*

Referências do capítulo

- Qubes OS — Glossary (net qube = None; qube offline). https://doc.qubes-os.org/en/latest/user/reference/glossary.html
- Qubes OS — Split GPG (vantagens sobre smart card; consentimento e notificação; limite da assinatura). https://doc.qubes-os.org/en/latest/user/security-in-qubes/split-gpg.html
- Qubes OS — External documentation: Split SSH (complementar). https://doc.qubes-os.org/en/latest/external/security-guidelines/split-ssh.html
- Qubes OS — CTAP proxy (proxy seguro; sem expor o navegador à pilha USB; U2F/FIDO2). https://doc.qubes-os.org/en/latest/user/security-in-qubes/ctap-proxy.html
- Qubes OS — How to copy and paste text (clipboard seguro a partir do vault). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-copy-and-paste-text.html

Módulo 15 — Ameaça física: criptografia, cold boot e evil maid

Todos os mecanismos vistos até aqui pressupõem um atacante remoto ou um software malicioso. Este módulo trata de outra categoria: o adversário que tem **acesso físico** ao seu computador — porque o roubou, o apreendeu, ou teve alguns minutos a sós com ele. Para quem está sob risco real, esse é frequentemente o cenário mais perigoso, e o Qubes oferece defesas específicas, com limites que é essencial compreender.

O ponto de partida é o mesmo do Módulo 4, e a documentação o repete: nenhum sistema operacional, nem mesmo o Qubes, pode ajudá-lo se você o instala em um hardware que já está comprometido. Software não conserta hardware adulterado. O que o Qubes pode fazer é proteger seus **dados** (com criptografia) e ajudá-lo a **detectar** adulterações no software de boot (com o Anti Evil Maid). São três os cenários a separar: dados em repouso, dados em uso e o ataque da "criada malvada".

Dados em repouso: a criptografia LUKS

Como visto no Módulo 5, o Qubes cifra o disco por padrão: por padrão, o Qubes OS usa LUKS/dm-crypt para cifrar tudo, exceto a partição **/boot**. Esse é o seu escudo para o cenário "máquina desligada e levada": sem a passphrase, o disco é ruído ilegível.

> *Aprofundamento (fonte: especificação LUKS / dm-crypt, fora da doc Qubes). Vale entender como o LUKS funciona, porque isso explica tanto sua força quanto seu limite. O disco é cifrado com uma **chave mestra** aleatória — não com a sua senha diretamente. A sua passphrase apenas desbloqueia um dos **key slots** do cabeçalho LUKS, que contêm a chave mestra cifrada. Para resistir a ataques de força bruta, a passphrase passa por uma **função de derivação de chave** custosa (PBKDF2 nas versões antigas, **Argon2** no LUKS2), calibrada para levar um tempo perceptível por tentativa. Por isso uma passphrase longa importa muito: cada caractere extra multiplica o custo de adivinhação. E por isso, também, trocar a senha é barato (reescreve só o key slot) e não exige recifrar o disco inteiro.*

Lembre o aviso do Módulo 5: se você esquecer a passphrase de criptografia, não há como recuperá-la. A força do LUKS é também sua intransigência. Mas atenção ao que ele **não** cobre, e é aqui que entra o ataque mais subestimado.

Dados em uso: o ataque de cold boot

O LUKS protege o disco em repouso. Mas, para você *usar* o computador, a chave mestra precisa estar carregada na **memória RAM** — é com ela que o sistema decifra os dados em tempo real. E é aí que mora a vulnerabilidade física mais traiçoeira.

> *Aprofundamento (fonte: pesquisa de segurança — cold boot attack, fora da doc Qubes). Ao contrário do que se imagina, a memória RAM não se apaga instantaneamente quando o computador é desligado: seu conteúdo persiste por segundos, e — se os chips forem resfriados (literalmente, com ar comprimido invertido ou nitrogênio) — por minutos. O **ataque de cold boot** explora isso: o atacante com a máquina ligada (ou suspensa) a reinicia abruptamente em um sistema mínimo que despeja a RAM, ou arranca os pentes de memória e os lê em outro equipamento. No despejo, ele procura a **chave mestra do***

> ***LUKS*** *— que estava ali, em claro, para o sistema funcionar. Com a chave, decifra o disco inteiro, contornando toda a criptografia. O alvo do ataque não é a senha; é a chave que a senha protegia, agora exposta na memória.*

A consequência prática é uma regra de postura que todo usuário de risco deve internalizar: **uma máquina ligada ou suspensa é uma máquina vulnerável**, mesmo com LUKS. O modo "suspender para a RAM" (suspend-to-RAM) é especialmente perigoso, porque mantém todas as chaves vivas na memória indefinidamente, com a tela apenas bloqueada. Por isso, em situações de risco — uma fronteira, uma viagem, sair de perto do equipamento — **desligar completamente** é muito mais seguro do que suspender: ao desligar, as chaves são removidas da RAM (e o Qubes faz a limpeza de memória dos qubes), e a janela do cold boot se fecha em segundos.

> *Reforço de privacidade. Para um jornalista cruzando uma fronteira ou um ativista cujo equipamento pode ser apreendido, essa distinção é literalmente a diferença entre dados protegidos e dados expostos. Um laptop fechado mas apenas suspenso, com LUKS, parece seguro — mas entrega a chave a quem souber fazer um cold boot. O hábito de* ***desligar de verdade*** *antes de qualquer situação de risco físico é uma das defesas de privacidade mais baratas e mais importantes que existem. Não confie no cadeado da tela; confie no botão de desligar.*

DMA versus remoção física: o que a VT-d cobre

Vale distinguir o cold boot de outro ataque à memória, porque eles pedem defesas diferentes. No Módulo 3B foi visto que a **VT-d/IOMMU** protege contra **ataques de DMA** — um dispositivo malicioso (ex.: por Thunderbolt/PCI) que tenta ler a RAM por acesso direto à memória. A IOMMU confina cada dispositivo ao espaço que o Xen lhe autoriza, barrando esse vetor.

Mas a VT-d **não** protege contra o cold boot, porque o cold boot não usa DMA: ele arranca os chips ou reinicia a máquina para ler a memória fisicamente, fora do controle do hypervisor. São vetores distintos: a IOMMU defende a RAM enquanto o sistema roda e controla os barramentos; o cold boot ataca a RAM justamente contornando o sistema. A única defesa robusta contra o cold boot é não deixar segredos na memória — ou seja, desligar.

A criada malvada e o Anti Evil Maid

O terceiro cenário é o "evil maid" (criada malvada): o atacante não rouba a máquina, mas tem acesso a ela enquanto você está fora (o clássico quarto de hotel). Ele não consegue ler o disco cifrado, mas pode **adulterar o software de boot** — a parte não cifrada (**/boot**) — para inserir um keylogger que capture sua passphrase de LUKS na próxima vez que você ligar. Depois, com a senha, ele volta e abre tudo.

A defesa do Qubes é o **Anti Evil Maid (AEM)**, que usa o hardware de confiança da máquina para detectar essa adulteração. A documentação informa os requisitos: o pacote atual exige uma interface TPM 1.2 e um motor Intel TXT funcional. O TPM (Trusted Platform Module) mede o estado do boot; se o software de boot foi alterado, a medição muda, e o AEM avisa você — tipicamente exibindo um segredo que só apareceria se o boot estivesse íntegro. Instala-se no dom0:

```
1 sudo qubes-dom0-update anti-evil-maid
```

*Aprofundamento. A ideia é a **atestação por TPM**: a cada boot, o TPM calcula hashes do firmware e do bootloader e os guarda em registradores (PCRs). O AEM "sela" um segredo a esses valores; se o boot for adulterado, os valores não batem, o segredo não é liberado, e você percebe que algo mudou antes de digitar sua passphrase. É detecção, não prevenção: o AEM não impede a adulteração, mas garante que você a note.*

O trade-off do AEM: um exemplo de modelo de ameaças

O AEM ilustra perfeitamente por que o Módulo 1 (modelo de ameaças) é a base de tudo. Em sua configuração padrão, o AEM exige anexar um pendrive diretamente ao dom0 — e isso colide com a regra de ouro do Módulo 12. A documentação coloca o dilema sem rodeios: cada usuário do Qubes precisa escolher entre, de um lado, proteger o dom0 de um pendrive potencialmente malicioso e, de outro, proteger o sistema de ataques evil maid. E lembra que nem mesmo pendrives novos, lacrados de fábrica, podem ser simplesmente presumidos "limpos".

A doc resolve isso não com uma resposta única, mas com modelo de ameaças. O exemplo é didático: um usuário que viaja com frequência com um laptop Qubes contendo dados sensíveis pode estar sob risco muito maior de ataques evil maid do que um usuário doméstico com um desktop Qubes fixo. A viajante frequente, mais exposta ao evil maid, pode racionalmente adotar o AEM; já o usuário doméstico, para quem o evil maid é improvável, pode julgar maior o risco do pendrive malicioso e optar por nunca anexar USB ao dom0. Não há resposta certa universal — há a resposta certa *para o seu modelo*.

Reforço de privacidade. Esse trade-off é um lembrete de que segurança física é inseparável do seu perfil de risco. Para uma fonte ou ativista que viaja com material sensível, o evil maid é uma ameaça concreta, e o AEM (somado a desligar a máquina) faz sentido. Para quem nunca tira o computador de casa, a complexidade pode não compensar. Reveja o seu Exercício 1: a decisão sobre o AEM, sobre desligar versus suspender, e sobre hardware com firmware aberto sai diretamente de quem são os seus adversários e do que eles podem fazer fisicamente.

Ocultar a existência do sistema: negação plausível

Há um degrau a mais de proteção física para quem enfrenta não só o roubo do equipamento, mas a **coação para entregá-lo funcionando**: esconder que o Qubes existe. A ideia, usada na prática por quem opera em risco alto, é instalar o Qubes em um **disco (ou SSD) separado** e deixar a máquina inicializar, por padrão, em um sistema comum e inofensivo — um "chamariz" (por exemplo, um Linux de uso corriqueiro). Só quem sabe do arranjo, ao interromper o boot e escolher o outro disco (uma combinação de teclas no menu de inicialização), chega ao Qubes. Para um observador casual — ou uma inspeção apressada —, o computador é apenas mais um laptop comum.

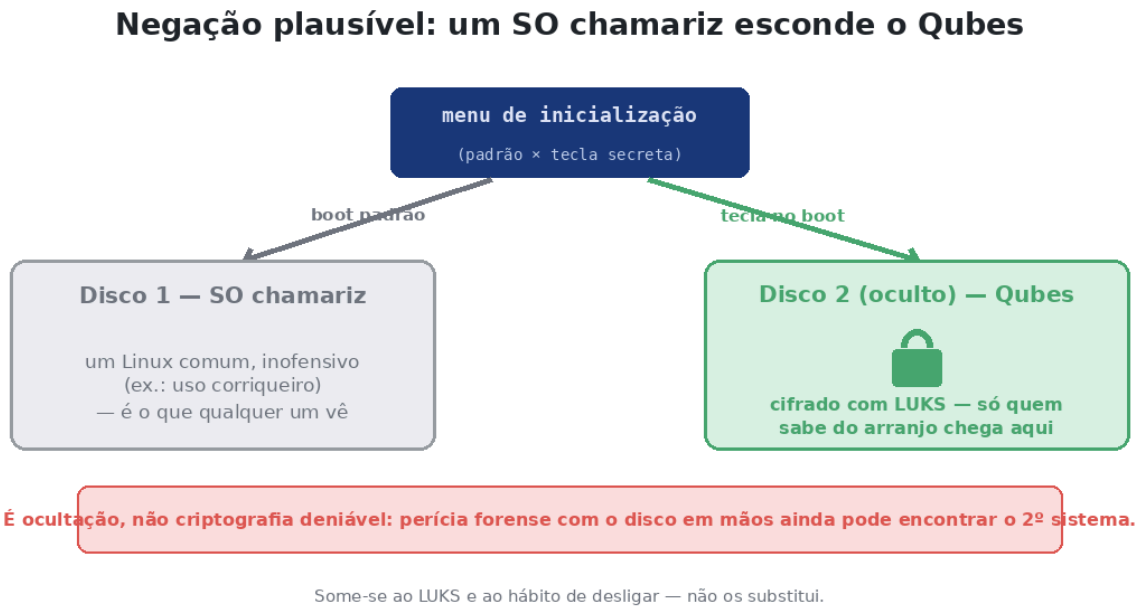


Figura: o boot padrão leva a um SO chamariz comum; só quem sabe do arranjo, com uma tecla no boot, alcança o disco oculto com o Qubes (cifrado com LUKS).

Esse arranjo se soma às defesas já vistas, não as substitui. O disco do Qubes continua cifrado com **LUKS** (dados em repouso protegidos), e continua valendo a regra do módulo: **desligar** antes de qualquer situação de risco. O chamariz apenas remove o Qubes da vista.

> *Reforço de privacidade. Para uma fonte, um jornalista ou um ativista, a diferença entre "seu disco está cifrado" e "não há sinal de que você use um sistema de compartimentalização" pode ser decisiva diante de quem pode exigir que você ligue e mostre a máquina. Mostrar um sistema comum, plausível e sem nada de mais é menos arriscado do que exibir uma tela de senha de disco cifrado que denuncia haver algo a esconder.*

Os limites, com franqueza. Isto é **ocultação**, não criptografia deniável: uma perícia cuidadosa pode encontrar a partição cifrada, o gerenciador de boot alterado ou vestígios no firmware, e concluir que há um segundo sistema. A técnica eleva o custo e derrota a inspeção superficial e a coação apressada — não um adversário forense com tempo e o disco em mãos. Combine-a, portanto, com o resto: LUKS forte, desligar, e um modelo de ameaças honesto sobre quem é o seu adversário e do que ele é capaz.

Laboratório 15 — Postura física e ciclo de vida da chave

Objetivo: consolidar a defesa física com uma demonstração conceitual e um checklist prático.

Passo 1 — **Verifique a criptografia.** No dom0, confirme que seu disco usa LUKS:

```
sudo blkid | grep crypto_LUKS
sudo cryptsetup luksDump /dev/<seu_dispositivo> | head -20
```

Observe o cabeçalho LUKS, os key slots e a função de derivação (KDF) em uso.

Passo 2 — **Ciclo de vida da chave (no papel).** Desenhe três estados: (a) máquina desligada → chave **não** está na RAM (segura contra cold boot); (b) máquina ligada/em uso → chave **na**

RAM (vulnerável a cold boot); (c) máquina suspensa → chave **ainda na RAM** (vulnerável). Marque qual estado você deve usar antes de uma situação de risco físico.

Passo 3 — **Checklist de postura física.** Monte o seu, por exemplo: passphrase LUKS longa e memorizada; **desligar** (não suspender) ao deixar o equipamento ou cruzar fronteiras; avaliar AEM conforme o risco de evil maid; considerar hardware com firmware aberto; nunca anexar USB ao dom0 sem necessidade.

Passo 4 — (Opcional, avançado) Se seu hardware tem TPM 1.2 e Intel TXT, leia a documentação do AEM e avalie instalá-lo, ponderando o trade-off do USB no dom0.

Passo 5 — Reflita e anote: por que um laptop apenas suspenso, com LUKS, ainda pode entregar seus dados? Qual ação simples fecha essa janela?

> *Atenção: estes comandos apenas inspecionam; não altere o cabeçalho LUKS nem os key slots — um erro pode tornar o disco irrecuperável.*

Referências do capítulo

- Qubes OS — Installation guide / Installation security (LUKS/dm-crypt; sem recuperação). https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/install-security.html
- Pesquisa: Halderman et al., "Lest We Remember: Cold Boot Attacks on Encryption Keys" (complementar). https://citp.princeton.edu/our-work/memory/
- Qubes OS — FAQ (VT-d/IOMMU contra DMA attacks). https://doc.qubes-os.org/en/latest/introduction/faq.html
- Qubes OS — Anti Evil Maid ("Security Considerations"; trade-off; viajante vs. usuário doméstico). https://doc.qubes-os.org/en/latest/user/security-in-qubes/anti-evil-maid.html

Módulo 16 — Backup, restauração e migração

Segurança não é só impedir o acesso indevido; é também garantir que você não perca o que é seu. Um disco que falha, um equipamento roubado, um erro que destrói um qube — todos são eventos de segurança, no sentido de disponibilidade. O Qubes trata o backup como cidadão de primeira classe, e a boa notícia, nas palavras da documentação, é que é fácil e seguro fazer backup e restaurar todo o sistema, além de migrar entre duas máquinas físicas.

As funções estão integradas ao Qube Manager, e há dois equivalentes de linha de comando: **qvm-backup** (criar) e **qvm-backup-restore** (restaurar). O ponto a guardar é que o backup do Qubes é, por construção, **cifrado e verificável** — ele nasce pensando no fato de que a cópia vai viver fora da máquina, em um lugar potencialmente menos seguro.

Criando um backup

O fluxo pelo Qube Manager: vá em Applications menu → System Tools → Backup Qubes. Na janela "Qubes Backup VMs", mova para a coluna **Selected** os qubes que deseja salvar (os que ficarem em **Available** não serão incluídos). Em seguida, escolha o destino — um qube em execução ou um dispositivo de armazenamento USB — e um diretório que **já exista** nele. Por fim, defina uma **passphrase**.

Um detalhe técnico importante: a passphrase fornecida é usada tanto para cifrar/decifrar quanto para verificar a integridade. A mesma frase secreta cifra o backup e garante que ele não foi adulterado. Por isso, escolha uma passphrase forte e guarde-a com o mesmo cuidado da sua senha de LUKS — sem ela, o backup é irrecuperável.

Há um aviso de segurança que vale destacar: existe a opção de salvar as configurações de backup, mas a doc adverte que salvar as configurações fará com que sua passphrase de backup seja gravada em texto claro no dom0, então considere o seu modelo de ameaças antes de marcar essa opção. Para um perfil de risco, não salve a passphrase em texto claro.

> *Reforço de privacidade. O backup cifrado é a extensão natural do raciocínio do Módulo 15: assim como o LUKS protege os dados no disco da máquina, a cifragem do backup protege seus dados na cópia — que muitas vezes acaba em um pendrive na gaveta, em um HD externo ou na nuvem, lugares fora do seu controle físico. Para quem guarda material sensível, um backup não cifrado é um vazamento esperando para acontecer: bastaria alguém encontrar o pendrive. Com a cifragem do Qubes, a cópia é tão ilegível quanto o disco original sem a passphrase.*

O caso especial do dom0

O dom0 tem uma regra própria que é fácil esquecer e cara descobrir tarde. Ao fazer backup do dom0 pela ferramenta, apenas o diretório home é incluído. Isso significa duas coisas: se você tem arquivos importantes fora de **/home** (por exemplo, políticas em **/etc/qubes**), copie-os para o home *antes* do backup; e os pacotes instalados pelo gerenciador de pacotes no dom0 não entram no backup — pacotes instalados no dom0 precisarão ser reinstalados manualmente em uma instalação nova. Planeje o dom0 como algo que você reconstrói, guardando apenas as configurações que personalizou.

Restaurando — e por que testar é obrigatório

Para restaurar: Applications menu → System Tools → Restore Backup. Você indica a origem (um qube ou o **sys-usb** com a mídia), o arquivo de backup e a passphrase. Há três opções úteis: **ignorar templates/netVMs ausentes** (restaura mesmo que o template original não exista, usando os padrões), **ignorar divergência de nome de usuário** (para o home do dom0, ao restaurar em um sistema com outro nome de usuário) e — a mais importante — **"Verify backup integrity, do not restore the data"**, que apenas confere se o arquivo está íntegro, sem restaurar.

A documentação insiste, com razão, em um hábito que muita gente ignora: faça uma restauração de teste do seu backup — um backup é inútil se você não consegue restaurar os dados a partir dele, e não há como ter certeza de que ele está bom até tentar restaurar. Faça a verificação de integridade após cada backup, e periodicamente faça uma restauração de teste real. Um backup nunca testado não é um backup — é uma esperança.

Migração entre máquinas e reversão de volumes

A mesma ferramenta de backup e restauração serve para **migrar** entre dois computadores físicos: você faz o backup na máquina antiga e o restaura na nova. Como o backup é cifrado, ele pode trafegar com segurança entre as duas, sem o problema do "caminho inseguro de transferência" discutido no Módulo 2.

Para necessidades mais granulares, há ainda o **volume backup/revert** (tópico avançado): a possibilidade de fazer cópia e reverter o volume de um qube específico — útil, por exemplo, para desfazer uma alteração em um qube sem restaurar o sistema inteiro.

Sem aprisionamento: o restore de emergência

Um último ponto, importante para quem desconfia de aprisionamento tecnológico: o formato de backup do Qubes é **aberto e documentado**, a ponto de existir um procedimento de "emergency restore" que recupera seus dados usando apenas ferramentas padrão de Linux (como **openssl**, **tar** e **gpg**), **sem precisar de um sistema Qubes funcionando**. Ou seja: mesmo que você não tenha um Qubes à mão, ou que a versão tenha mudado, seus dados não ficam reféns. Para quem leva privacidade a sério, essa transparência é uma garantia a mais — você sempre poderá abrir o seu próprio backup.

Laboratório 16 — Backup e restauração ponta a ponta

Objetivo: criar um backup cifrado, verificar sua integridade e entender o ciclo completo.

Passo 1 — Abra **Backup Qubes** (System Tools). Selecione um ou dois qubes pequenos (ex.: **personal**) para a coluna Selected.

Passo 2 — Escolha um destino — um diretório existente em outro qube ou em um pendrive anexado — e defina uma **passphrase** forte. **Não** marque a opção de salvar a passphrase. Execute o backup.

Passo 3 — Abra **Restore Backup**. Aponte para o arquivo gerado, informe a passphrase e marque **"Verify backup integrity, do not restore the data"**. Confirme que a verificação passa.

Passo 4 — (Opcional) Faça uma restauração real para confirmar que os dados voltam corretamente.

Passo 5 — Reflita e anote: por que um backup nunca testado não pode ser considerado confiável? E por que a cifragem do backup é tão importante quanto a do disco (LUKS) do Módulo 15?

> *Atenção: trate a passphrase do backup como segredo crítico — sem ela, o backup é irrecuperável, exatamente como a senha do LUKS.*

Referências do capítulo

- Qubes OS — How to back up (seleção de VMs; destino; passphrase para cifra e integridade; aviso de plaintext). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-back-up-restore-and-migrate.html
- Qubes OS — Volume backup and revert. https://doc.qubes-os.org/en/latest/user/advanced-topics/volume-backup-revert.html
- Qubes OS — Backup emergency restore. https://doc.qubes-os.org/en/latest/user/how-to-guides/backup-emergency-restore-v4.html

Módulo 17 — Atualização e manutenção do sistema

No Módulo 9 ficou visto como instalar e atualizar software. Este módulo trata da manutenção como disciplina: o ciclo de atualizações, como o Qubes mantém o dom0 seguro mesmo precisando de updates da internet, como reinstalar um componente comprometido e como manter o sistema saudável. A premissa, repetida pela documentação, é que atualizar com frequência é uma das melhores formas de se manter seguro contra novas ameaças. E que, em incidentes sérios, o projeto emite um **Qubes Security Bulletin (QSB)** — é muito importante ler cada novo QSB e seguir quaisquer instruções ao usuário que ele contenha.

Como o dom0 atualiza sem ter rede

Aqui está um dos pontos mais engenhosos do Qubes, e ele resolve um paradoxo aparente: o dom0 não tem rede (Módulos 3B e 6), então como ele se atualiza? A documentação explica o mecanismo, projetado para minimizar a quantidade de entrada não confiável processada pelo software do dom0.

O processo: um **UpdateVM** (por padrão, o mesmo qube do firewall, conectado à rede) baixa os pacotes de atualização. O dom0 nunca toca a internet; ele apenas pede ao UpdateVM que busque os arquivos. E aqui está a chave da segurança — a documentação assume o pior: parte-se do princípio de que esse script pode ser comprometido e baixar downloads maliciosamente adulterados, e isso não é um problema, porque o dom0 verifica depois as assinaturas digitais das atualizações.

Em seguida, os pacotes são entregues ao dom0 via qrexec, e o serviço **qubes-receive-updates** processa a entrada não confiável vinda do UpdateVM e então verifica a assinatura digital de cada arquivo. Só depois de verificada a assinatura é que os pacotes passam a ser confiáveis. É a mesma filosofia da verificação da ISO (Módulo 4), aplicada às atualizações: o dom0 desconfia de tudo o que vem da rede e confia apenas no que está assinado pelas chaves do projeto.

> *Reforço de privacidade. Esse desenho protege a parte mais crítica do sistema sem expô-la. Mesmo que um adversário comprometa o qube que baixa as atualizações — ou intercepte a conexão —, ele não consegue injetar código malicioso no dom0, porque as assinaturas não baterão. E, combinando com o Módulo 11, você pode configurar o UpdateVM para baixar via **sys-whonix**, de modo que nem mesmo quando e o que você atualiza fique visível para um observador de rede. Atualizar deixa de ser um momento de exposição.*

A ferramenta e os reinícios necessários

A forma correta de atualizar é a **Qubes Update** (ícone na bandeja quando há updates) ou, no terminal do dom0, **qubes-dom0-update** (para o dom0) e **qubes-vm-update** (para os templates). Lembre do Módulo 9: **não** use **dnf update/apt update** diretos — eles contornam as proteções do Qubes.

Dois reinícios são essenciais e fáceis de esquecer:

- O dom0 deve ser reiniciado após todas as atualizações de Xen e de kernel (e, em sistemas Intel, após updates de **microcode_ctl**; em AMD, de **linux-firmware**). Atualizações do hypervisor e do kernel só passam a valer após o reboot.

- Após atualizar um template, primeiro desligue o template e depois reinicie todos os qubes em execução baseados nesse template. A ferramenta tenta fazer isso por você, mas salve seu trabalho antes. Sem reiniciar, os qubes continuam rodando a versão antiga do sistema (Módulo 7).

O Whonix se atualiza pelo mesmo fluxo; quando feito por Tor, a doc lembra que isso pode demorar bastante.

Grandes atualizações: upgrade in-place e de templates

Dois tipos de atualização vão além do dia a dia e merecem nota. O **upgrade in-place** entre versões do Qubes (por exemplo, de uma 4.x para a seguinte) não é reinstalação: há um procedimento oficial, conduzido por etapas no dom0 com a ferramenta **qubes-dist-upgrade**, que migra o sistema preservando seus qubes — leia o guia da versão-alvo antes e **faça backup** (Módulo 16) primeiro. Já o **upgrade de template** troca a base (por exemplo, uma versão nova do Fedora ou do Debian) quando a atual se aproxima do fim de suporte: instala-se o template novo, migram-se os app qubes para ele e removem-se os antigos. Em ambos os casos, o hábito é o mesmo: backup antes, ler as notas da versão, e reiniciar o que for preciso.

Reinstalando um template comprometido

E se você suspeitar que um template foi corrompido ou comprometido — por exemplo, após instalar nele um pacote de origem duvidosa (lembre da herança de confiança do Módulo 7)? O Qubes permite reinstalá-lo do zero. Nas palavras da documentação: se você suspeita que seu template está quebrado, mal configurado ou comprometido, pode reinstalar qualquer template que tenha sido instalado a partir do repositório do Qubes. O comando, no dom0:

```
sudo qubes-dom0-update --action=reinstall qubes-template-fedora-XX
```

Um detalhe operacional: os qubes que estão usando o template reinstalado não são afetados até serem reiniciados. A reinstalação troca o molde; os app qubes só herdam o template limpo ao reiniciarem.

> *Reforço de privacidade. A capacidade de reinstalar um template é uma ferramenta de recuperação valiosa. Se você desconfia que um template foi comprometido, reinstalá-lo (e reiniciar os qubes baseados nele) restaura um sistema limpo para todos os seus filhos de uma vez — sem precisar recriar cada qube. Combine com a lição do Módulo 7: se a suspeita for séria, considere também que dados em **/home** dos app qubes não são tocados pela reinstalação do template, então avalie quais qubes podem precisar de atenção adicional.*

Mantendo o sistema saudável

Além das atualizações, alguns hábitos de manutenção mantêm o sistema enxuto e seguro:

- **Espaço em disco.** O widget Qubes Disk Space (Módulo 6) avisa quando o armazenamento aperta. Remova qubes e templates que você não usa mais — cada um é também superfície de manutenção.
- **Menos é mais.** Templates que não têm nenhum qube filho online podem nem receber notificação de atualização; mantenha apenas os que você de fato usa, e atualize-os.

- **Acompanhe os updates.** Na ferramenta Qubes Update, a coluna "last checked" mostra quando cada qube verificou atualizações pela última vez. Qubes marcados como "OBSOLETE" indicam versões sem suporte — migre-os.
- **Reduza superfície.** Periodicamente, revise quais aplicativos estão instalados nos templates; desinstalar o que não se usa é reduzir a superfície de ataque (Módulo 1) em todos os qubes derivados.

Mantendo-se informado

A manutenção também é informacional. Acompanhe os **Qubes Security Bulletins (QSBs)**, que descrevem vulnerabilidades e as ações necessárias, e o **Qubes Security Pack (qubes-secpack)**. Na maioria das vezes, atualizar normalmente já aplica as correções; mas alguns QSBs pedem ações específicas. Para quem opera sob risco, ler os QSBs é parte da rotina de segurança, tanto quanto rodar as atualizações. As fontes da comunidade voltam no Módulo 19.

Laboratório 17 — Ciclo de atualização e manutenção

Objetivo: executar o ciclo completo de atualização e praticar a manutenção.

Passo 1 — Abra a ferramenta **Qubes Update**. Observe a lista de qubes/templates, a coluna "last checked" e quais têm updates disponíveis ("YES"/"NO"/"OBSOLETE").

Passo 2 — Selecione o dom0 e um template e execute as atualizações. Observe que o download passa pelo UpdateVM e que o dom0 verifica as assinaturas.

Passo 3 — Após o término, verifique se há aviso para reiniciar (especialmente se houve update de Xen/kernel). Se houver, planeje o reboot do dom0; reinicie os qubes baseados no template atualizado.

Passo 4 — **Manutenção.** Abra o widget Qubes Disk Space e verifique o uso. No Qube Manager, identifique qubes ou templates que você não usa e considere removê-los (não remova qubes de sistema).

Passo 5 — Reflita e anote: por que o fato de o dom0 verificar assinaturas torna seguro deixar o download das atualizações nas mãos de um qube com rede (e potencialmente comprometido)? Como isso se parece com a verificação da ISO no Módulo 4?

> *Atenção: reinstalar ou remover templates afeta todos os qubes baseados neles. Confirme as dependências (coluna TEMPLATE no **qvm-ls**) antes de qualquer remoção.*

Referências do capítulo

- Qubes OS — How to update (Qubes Update; reiniciar dom0 após Xen/kernel; reiniciar qubes após template). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-update.html
- Qubes OS — Dom0 secure updates (UpdateVM baixa; dom0 verifica assinaturas). https://doc.qubes-os.org/en/latest/developer/system/dom0-secure-updates.html
- Qubes OS — How to reinstall a template (suspeita de comprometimento; comando; reiniciar). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-reinstall-a-template.html
- Qubes OS — Upgrade guides (upgrade in-place com qubes-dist-upgrade; upgrade de templates). https://doc.qubes-os.org/en/latest/user/downloading-installing-upgrading/

Módulo 18 — Projetando seu esquema de compartimentalização

Você já conhece todas as peças: templates e app qubes, descartáveis, qubes de sistema, vault, Whonix, dispositivos, segredos. Este módulo é a oficina onde você as combina em um esquema coerente — o seu. É também o reencontro com o Exercício 1 (Módulo 1): aquele modelo de ameaças que você escreveu lá no início agora vira um desenho concreto de quantos qubes criar, com que propósito e com que nível de confiança.

A documentação começa desarmando a ansiedade da página em branco: cada qube é, em essência, um compartimento seguro, e você pode criar quantos quiser; são como blocos de Lego, no sentido de que você monta o que bem entender. E é direta sobre não existir gabarito: ninguém pode lhe dizer exatamente como organizar seus qubes, porque não há uma única resposta certa para essa pergunta — depende das suas necessidades, desejos e preferências. O que dá para fazer é estudar um caso real próximo do público deste livro e extrair padrões reutilizáveis.

Princípios de organização

Antes do caso, três eixos para você pensar a sua divisão:

- **Por nível de confiança.** Separe o que é sensível do que é arriscado. Um qube **untrusted** para navegação aleatória; um **vault** offline para segredos; qubes de trabalho no meio. A cor (Módulo 6) sinaliza isso visualmente.
- **Por identidade/contexto.** Não misture papéis. Trabalho, pessoal, anônimo e cada cliente/projeto/fonte em compartimentos distintos, para que um vazamento não ligue uma coisa à outra. Para quem precisa de anonimato, separar a identidade real da anônima é inegociável (Módulo 11).
- **Por finalidade técnica.** Qubes de sistema (rede, USB), qubes de aplicação, vaults offline, gateways (Whonix). Um esquema de nomes consistente (ex.: **clienteA-trabalho**, **fonte-tips**) mantém tudo organizado.

A documentação dá um conforto importante: a bagunça pode existir *dentro* de um qube sem ameaçar o todo. Uma usuária do exemplo sente-se tranquila por saber que as coisas podem estar bagunçadas e desorganizadas dentro de um qube enquanto a vida digital como um todo permanece bem organizada. A compartimentalização compra essa liberdade.

Estudo de caso: Bob, o jornalista investigativo

O exemplo mais próximo do público deste livro é o de Bob, que é frequentemente forçado a lidar com arquivos suspeitos, muitas vezes de fontes anônimas. Ele recebe anexos que dizem ser uma denúncia, mas podem ser malware. Bob não é técnico — só quer revelar a verdade — mas sabe que a natureza do seu trabalho pode torná-lo um alvo e que precisa proteger suas fontes, seus colegas, sua família e a si mesmo. O esquema dele, num laptop dedicado só ao trabalho, é uma aula de compartimentalização aplicada à privacidade:

- **Um qube offline para escrever.** Roda apenas o LibreOffice Writer, sem rede. É onde Bob redige, ao lado da janela com a pesquisa.
- **Múltiplos qubes de e-mail**, baseados em template **minimal** com Thunderbird. Um para o público geral, outro para o editor e colegas. Ambos configurados para abrir

todos os anexos em **disposables offline**, para o caso de um anexo conter um beacon que tente "telefonar para casa".

- **Qubes Whonix.** O **sys-whonix** para acesso Tor, e disposables **anon-workstation** para pesquisar com o Tor Browser. A razão é puro modelo de ameaças: ele não quer que suas requisições de rede sejam rastreadas até o IP do trabalho ou de casa. Ele tem ainda um disposable template Whonix dedicado a **receber denúncias anonimamente via Tor**, porque alguns whistleblowers de alto risco não podem se arriscar com nenhuma outra forma de comunicação.
- **Dois qubes para Signal.** Um ligado ao número pessoal (contatos conhecidos) e outro, um número público, uma via adicional para que fontes o contatem de forma confidencial.
- **Vários vaults de dados,** offline. Como o material vem de fontes desconhecidas, Bob não tem certeza se seria seguro colocar tudo no mesmo vault, então cria vaults diferentes (em geral, um para cada matéria ou tema). Um cofre por investigação.
- **Um qube de VPN** para acessar os recursos da redação remotamente.
- **Um vault de gerenciador de senhas**, offline, de onde ele copia credenciais com o clipboard seguro.

Reforço de privacidade. Repare como cada decisão de Bob deriva diretamente de uma ameaça concreta à privacidade — a dele e a das fontes. Anexos abertos em disposables offline impedem que um "beacon" denuncie que ele abriu o documento. A pesquisa via Whonix impede que o alvo da investigação descubra que está sendo investigado. Vaults separados por história impedem que o comprometimento de um material contamine os outros. Dois Signals separam contatos confiáveis de fontes anônimas. Nenhuma dessas escolhas é "técnica por capricho": cada uma protege uma pessoa real de um risco real. É assim que você deve pensar o seu próprio esquema.

Padrões reutilizáveis

Do caso de Bob (e dos demais perfis da documentação) emergem padrões que você pode adaptar:

- **Vault offline para o que importa.** Documentos, chaves, senhas — sempre em qubes sem rede (**net qube = None**). Crie vaults separados por tema quando a confiança das fontes variar.
- **E-mail que abre anexos em disposable.** Configure o cliente para abrir todo anexo em um descartável (de preferência offline). Vetor mais comum de ataque, neutralizado por padrão.
- **Pesquisa e contato sensível via Whonix.** Use disposables Whonix para investigar e para receber material anônimo. Anonimato é propriedade da arquitetura, não do seu cuidado momentâneo.
- **Identidades separadas em qubes separados.** Trabalho, pessoal, anônimo; números públicos e privados; cada cliente/fonte isolado. Nunca cruze.
- **Template minimal para qubes de propósito único.** Menos software, menos superfície (Módulo 9).
- **Firewall restritivo por qube.** Limite cada qube ao que ele precisa alcançar (Módulo 10).
- **Microfone/câmera sob demanda.** Anexe só durante o uso (Módulo 12) — sem confiar no botão de "mudo" de nenhum app.

- **Descartar e restaurar.** Ao terminar um projeto, faça backup do conjunto e remova os qubes; restaure do backup se precisar de novo.

As três regras simples

O colega que ajudou Bob a montar o sistema lhe deixou três regras que resumem boa parte do curso, e que valem para qualquer esquema, por mais simples que seja: não copie nem mova textos ou arquivos de qubes menos confiáveis para qubes mais confiáveis; atualize o sistema quando for solicitado; e faça backups regulares.

São, respectivamente, a regra de direção (Módulo 8), a disciplina de atualização (Módulos 9 e 17) e o backup (Módulo 16). Se um usuário não técnico como Bob consegue operar com segurança seguindo apenas essas três, é porque a arquitetura faz o trabalho pesado — desde que você respeite esses três hábitos.

> *Reforço de privacidade. Essas três regras protegem privacidade tanto quanto segurança. A regra de direção evita que material não confiável contamine seus compartimentos sensíveis; atualizar fecha brechas que poderiam expor você; e o backup cifrado garante que a perda do equipamento não vire perda — nem vazamento — dos dados. Internalize-as como hábito, não como tarefa.*

Oficina 18 — Desenhe e implemente o seu esquema

Objetivo: transformar o seu modelo de ameaças (Exercício 1) em um esquema concreto de qubes.

Passo 1 — **Retome o Exercício 1.** Releia o que você quer proteger, de quem, com que gravidade e probabilidade. Liste seus "papéis" digitais (trabalho, pessoal, anônimo, finanças, fontes, etc.).

Passo 2 — **Mapeie papéis em qubes.** Para cada papel, defina: nome, cor (nível de confiança), template (padrão ou minimal), net qube (rede normal, Whonix, ou None), e se deve ser disposable. Marque quais segredos vão para vaults offline e quais identidades precisam de anonimato (Whonix).

Passo 3 — **Desenhe o diagrama.** Em papel ou ferramenta, desenhe os qubes e as conexões: quem usa quem como net qube, qual vault serve qual frontend (Split-GPG), onde os anexos abrem (disposable). Inclua sys-net, sys-firewall, sys-usb e os vaults.

Passo 4 — **Implemente o núcleo.** Crie ao menos: um vault offline para segredos, um qube de trabalho com firewall restrito, um qube **untrusted**, e (se precisar de anonimato) confirme o anon-whonix. Use **qvm-create**, defina **netvm** e cores com **qvm-prefs**.

Passo 5 — **Valide com as três regras.** Confirme que você sabe, para o seu esquema: qual cópia é segura (direção), como vai atualizar, e como vai fazer backup. Anote os qubes que conterão dados críticos a incluir no backup.

> *Atenção: comece simples. É melhor um esquema modesto que você mantém do que um complexo que você abandona. Cresça à medida que a necessidade aparecer.*

Referências do capítulo

- Qubes OS — How to organize your qubes (esquemas de nome; ordem entre qubes). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-organize-your-qubes.html

- Qubes OS — How to copy and paste text / How to update / How to back up. https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-back-up-restore-and-migrate.html

Módulo 19 — Solução de problemas, comunidade e próximos passos

Nenhum sistema é livre de percalços, e o Qubes — por sua complexidade — tem os seus. A boa notícia é que a maioria dos problemas é conhecida e documentada. A documentação mantém páginas de troubleshooting por categoria: instalação, UEFI, USB, GUI, suspender/retomar, disco, PCI, rede, atualização, entre outras. Antes de entrar em pânico, identifique a categoria do seu problema e consulte a página correspondente.

Alguns problemas são especialmente comuns e vale conhecer de antemão:

- **Rede caiu após o sistema acordar.** Como visto no Módulo 10, o **sys-net** pode travar ao retornar do suspend (problemas de driver Wi-Fi). Em geral o Qubes reconecta sozinho; se não, há um comando no dom0 para restaurar o vínculo do **sys-firewall**.
- **Suspender/retomar.** Problemas de retorno do sono são frequentes e dependentes de hardware — há uma página dedicada.
- **USB não funciona.** Lembre que dispositivos passam pelo **sys-usb** (Módulo 12); muitos "problemas" são, na verdade, dispositivos não anexados.
- **Atalho de aplicativo sumiu / GUI estranha.** Há páginas específicas para o menu de aplicativos e para a GUI.

O método geral, recomendado pela documentação, é pesquisar no issue tracker, pois pode já haver uma issue aberta ou fechada sobre o seu problema, e buscar também no fórum. Muitas vezes a solução — ou um contorno — já existe.

Buscar ajuda — com segurança

A comunidade do Qubes é ativa e acolhedora — está ali para ajudar. Os dois principais canais são o **Qubes Forum** e a lista **qubes-users**. Um pedido prático da doc: não pergunte nos dois ao mesmo tempo. E, antes de perguntar, pesquise — issue tracker, fórum e arquivos da lista costumam já ter a resposta.

Mas, sendo um curso de privacidade, o ponto mais importante aqui é *como* pedir ajuda com segurança. A documentação alerta: as listas de discussão e o fórum do Qubes são abertos ao público; o conteúdo é rastreado por mecanismos de busca e arquivado por serviços de terceiros, fora do controle do projeto. Não envie nem poste nada que você não se sinta confortável de ver discutido publicamente; se a confidencialidade for uma preocupação, use cifragem PGP em um e-mail fora da lista.

> *Reforço de privacidade. Para o público deste livro, esse aviso é crítico. Ao pedir ajuda, é tentador colar logs, nomes de qubes, configurações de rede — informações que podem revelar quem você é, o que você faz e contra quem se protege. Antes de postar, **sanitize**: remova nomes reais, IPs, identificadores e qualquer pista do seu contexto. Lembre que o que vai para um fórum público é indexado e arquivado para sempre, fora do seu controle. Se o assunto for sensível, prefira um canal cifrado. A higiene de privacidade vale também — e especialmente — na hora de buscar suporte.*

Há ainda uma dica de confiança elegante: como toda contribuição ao projeto é assinada criptograficamente, você pode preferir conselhos de quem tem reputação construída. A doc observa que é improvável que pessoas que trabalharam duro para construir boas reputações

arrisquem dar conselhos maliciosos em mensagens assinadas, e que qualquer um pode verificar a assinatura. Desconfie de instruções anônimas que peçam para você baixar algo estranho ou rodar comandos no dom0.

O que o Qubes não protege

Ao longo do curso, sempre que uma defesa foi apresentada, seu limite foi apontado. Este é o momento de reunir esse rigor, porque ele é parte da competência de quem usa o Qubes a sério. O sistema se descreve como "reasonably secure" (razoavelmente seguro), não perfeito — e conhecer as fronteiras o torna um usuário melhor.

O exemplo mais instrutivo está na documentação sobre vazamentos de dados, que afirma de saída que o firewall no Qubes não foi pensado para ser um mecanismo de prevenção de vazamentos. Por quê? Porque o Qubes não consegue impedir que um atacante que comprometeu um app qube com regras restritivas de firewall vaze dados, por canais ocultos cooperativos, através de outro app qube comprometido com regras de firewall não restritivas. Em outras palavras, se dois dos seus qubes estiverem comprometidos, malware cooperante pode mover dados entre eles por **canais ocultos (covert channels)** — a doc cita o exemplo de usar o **cache da CPU** como canal, com banda de algumas dezenas de bits por segundo. É um ataque sofisticado, que exige malware escrito especificamente contra o Qubes, mas que pode ser possível.

Há também os **canais laterais (side channels)**: malware tentando inferir segredos de outra VM observando, por exemplo, padrões de tempo de uma operação criptográfica. A doc nota que tais ataques são descritos na literatura acadêmica, mas que é duvidoso que tivessem sucesso na prática em um sistema geral e ocupado como o Qubes. E faz uma ressalva que ecoa o Módulo 2: máquinas fisicamente isoladas (air-gapped) não são necessariamente imunes a esse problema — canais ocultos podem usar até microfone e alto-falantes.

Some-se a isso tudo o que já foi visto: 0-days no próprio Xen (tratados via XSAs, Módulo 3B); o cold boot e a remoção física de RAM (Módulo 15); a confiabilidade da assinatura que o Split-GPG não resolve (Módulo 14); o anonimato que só o Whonix entrega e que você ainda pode estragar revelando-se (Módulo 11); e o hardware que, se já chegou comprometido, nenhum software salva (Módulo 4).

> *Reforço de privacidade. Reconhecer esses limites não enfraquece sua privacidade — fortalece. Um usuário que sabe que canais ocultos exigem **dois** qubes comprometidos entende por que vale a pena não abrir tudo no mesmo lugar. Um usuário que sabe que o Whonix esconde o IP mas não o que ele revela não faz login na conta real dentro do qube anônimo. A segurança real não vem de acreditar que se está invulnerável, mas de saber exatamente onde estão as bordas e operar dentro delas com disciplina. Essa é, talvez, a lição mais importante do curso.*

Mantendo-se atualizado e seguro

A segurança é um processo, não um estado. Acompanhe os **Qubes Security Bulletins (QSBs)** — leia cada um e aplique as ações recomendadas (Módulo 17). Mantenha o sistema atualizado, faça backups, e revise periodicamente o seu esquema de compartimentalização à medida que suas necessidades mudam. As três regras do Módulo 18 — direção da cópia, atualizar, fazer backup — são o mínimo que sustenta tudo.

Próximos passos

Você concluiu o essencial. A partir daqui, o caminho é de aprofundamento conforme a sua necessidade:

- **Tópicos avançados:** templates minimal customizados, bind-dirs, standalones e HVMs, políticas RPC, Salt para automação, USB qubes — tudo na seção Advanced Topics da documentação.
- **Whonix a fundo:** se anonimato é central para você, a documentação do projeto Whonix é leitura obrigatória (fingerprinting, deanonymization, uso correto do Tor).
- **Hardware:** se o seu risco justifica, estude hardware certificado e firmware aberto (coreboot/Heads).
- **Contribuir:** o Qubes é comunitário; reportar bugs, melhorar a documentação ou ajudar outros no fórum fortalece o ecossistema do qual você depende.

Laboratório 19 — Resolver, buscar e fechar o ciclo

Objetivo: praticar a resolução de um problema comum e a busca segura por ajuda.

Passo 1 — **Reconexão de rede.** Force o desligamento do **sys-net** (simulando um crash) e observe o efeito nos qubes acima. Em seguida, restaure a conexão pelo dom0 (comando da página de firewall) e confirme que a rede voltou.

Passo 2 — **Busca no issue tracker/fórum.** Escolha um sintoma real ou hipotético (ex.: "sys-usb não inicia") e pratique buscá-lo no issue tracker e no Qubes Forum. Observe como issues fechadas frequentemente já trazem a solução.

Passo 3 — **Encontre um QSB.** Localize a lista de Qubes Security Bulletins e leia um recente. Identifique se ele exigiu ação do usuário além de atualizar.

Passo 4 — **Sanitize um pedido de ajuda.** Escreva um pedido de suporte fictício sobre um problema seu e, em seguida, revise-o removendo qualquer informação que revele sua identidade ou contexto. Compare as duas versões.

Passo 5 — **Fechamento.** Revisite o seu esquema da Oficina 18 e a sua resposta do Laboratório 3B (qual domínio você atacaria primeiro). Com tudo o que aprendeu, reescreva essa análise citando quais mecanismos e quais limites se aplicam.

> *Atenção: ao buscar ajuda de verdade, nunca rode no dom0 comandos sugeridos por fontes não confiáveis, e nunca poste logs sem sanitizá-los.*

Uma palavra de encerramento

Você começou este curso com uma premissa desconfortável: todo software tem bugs, e um dia algum deles será explorado. Em vez de prometer que isso nunca aconteceria, o Qubes — e este curso — escolheu uma resposta mais franca e mais forte: confinar, controlar e conter o dano. Ao longo de dezenove módulos, você viu como essa ideia se materializa em engenharia: um hypervisor pequeno, um dom0 sem rede, qubes isolados por hardware, descartáveis que somem, vaults offline, Tor imposto no gateway, e a disciplina de conferir a cor antes de digitar uma senha.

Mais importante, você aprendeu a pensar como quem se protege de verdade: a partir de um modelo de ameaças, com expectativas realistas sobre o que cada defesa cobre e o que ela

não cobre. Essa mentalidade vale além do Qubes. Que ela o sirva — e proteja quem depende de você.

Referências do capítulo

- Qubes OS — Troubleshooting (índice de páginas por categoria). https://doc.qubes-os.org/en/latest/user/troubleshooting/installation-troubleshooting.html
- Qubes OS — Firewall (reconectar após reboot do net qube). https://doc.qubes-os.org/en/latest/user/security-in-qubes/firewall.html
- Qubes OS — Help, support, mailing lists, and forum (canais; staying safe; conselhos assinados). https://doc.qubes-os.org/en/latest/introduction/support.html
- Qubes OS — Data leaks (firewall não é mecanismo anti-vazamento; canais ocultos e laterais). https://doc.qubes-os.org/en/latest/user/security-in-qubes/data-leaks.html
- Qubes OS — How to update (QSBs). https://doc.qubes-os.org/en/latest/user/how-to-guides/how-to-update.html
- Qubes OS — Documentation (índice; tópicos avançados). https://doc.qubes-os.org/en/latest/index.html
- Whonix — Documentation (complementar). https://www.whonix.org/wiki/Documentation