HACKER

ENTRE A LUZ E AS TREVAS

IDENTIDADE · ANONIMATO · SOBREVIVÊNCIA

o caminho do hacker é o conhecimento

Nao.Importa.Web

Livreebooks.com.br

Nao.Importa.Web

Hacker: Entre a Luz e as Trevas

Identidade, Anonimato e Sobrevivência

2ª edição
Brasil
Livreebooks
2026

https://www.livreebooks.com.br

Nao.Importa.Web
Hacker: Entre a Luz e as Trevas : Identidade, Anonimato e Sobrevivência / Nao.Importa.Web
Brasil : Livreebooks, 2026.

1. Primeiros Passos de um Hacker. 2. Identidade Digital e Autenticação. 3. Anonimato. 4. Privacidade. 5. Tor: o Onion Router. 6. Whonix. 7. VPN Privada. 8. Rede I2P.

CDD 005.8 CDU 004.056

Ficha catalográfica

Aos que caminham no vale da sombra e não temem; aos que trocaram a vontade de ter uma marca pelo direito a um futuro seguro; e a todos que entenderam que privacidade não é ter algo a esconder — é ter algo a proteger.

Agradecimentos

Nenhum hacker se faz sozinho. Este livro deve o que sabe a uma linhagem que começou muito antes de mim: dos encontros do Homebrew Computer Club, na garagem onde a computação pessoal deixou de ser privilégio de poucos e virou cultura, aos cypherpunks que transformaram criptografia em liberdade. Agradeço a quem constrói e mantém as ferramentas que nos guardam anônimos — o Tor, o Whonix, a I2P, as criptomoedas de privacidade — e aos companheiros de trincheira, em especial ao Petrov e ao Jhon China Team, que nunca me deixaram esquecer que estudar é resistir. E a você, que abriu este livro: que ele o mantenha vivo e livre.

As empresas gastam milhões em firewalls, criptografia e dispositivos de acesso seguro, e é dinheiro desperdiçado — porque nenhuma dessas medidas cuida do elo mais fraco da corrente de segurança: as pessoas.

— Kevin Mitnick, A Arte de Enganar

Apresentação

Este não é um livro de ferramentas; é um livro de mentalidade. Ferramentas mudam a cada temporada, mas a forma de pensar do hacker atravessa gerações. Ela começou muito antes das telas que temos hoje — nas reuniões do Homebrew Computer Club, em 1975, onde um punhado de entusiastas provou que a tecnologia podia ser tirada das mãos das corporações e devolvida às pessoas. E ganhou um capítulo incômodo com Kevin Mitnick, que mostrou ao mundo, da pior forma para as empresas, que o elo mais fraco de qualquer sistema nunca foi o silício: é o ser humano. É dessa linhagem — construtores e trapaceiros, luz e trevas — que este livro se reconhece herdeiro.

A tese que sustenta cada página é simples e desconfortável: nenhum curso forma um hacker, e o caminho do conhecimento não se compra empacotado. Antes de qualquer técnica ofensiva vem a sobrevivência — o anonimato, a identidade, a criptografia —, porque de nada adianta saber atacar se você não sabe existir sem ser encontrado. Por isso o subtítulo desta obra é _entre a luz e as trevas_: não há uma dicotomia limpa entre o bem e o mal, entre o white hat e o black hat; há um _entre_, e é nesse território que quase todo hacker de verdade vive.

Leia com o teclado por perto e a mente aberta, mas sem ilusões. Cada técnica aqui pressupõe autorização e responsabilidade: sem elas, não há hacking ético — há crime. O que se pretende deixar em você não é uma coleção de comandos decorados, e sim uma forma de raciocinar sobre risco, poder e liberdade. Se ao fim destas páginas você for um pouco mais difícil de rastrear e um pouco mais difícil de enganar, o livro terá cumprido seu papel.

Lista de figuras

Primeiros Passos de um Hacker

Nestes primeiros capítulos vou me colocar na narrativa e conjugar o verbo na primeira pessoa. Vou me adicionando ao texto e, naturalmente, dando uma opinião clara — em parte para reduzir o número de pessoas que tentam entrar no mundo hacker sem nenhuma aptidão para a área. Este é um capítulo que precisa descrever bem o que é esse mundo e como os hackers se posicionam nele. Ele não é sobre ferramentas nem sobre comandos: é sobre mentalidade, sobre onde você vai acabar se encaixando e sobre a única coisa que ninguém vende num curso — sobreviver a esse caminho.

Nenhum curso vai te formar

Para começar, vou dizer uma coisa que desagrada muita gente: nenhum curso hacker vai lhe formar ou lhe habilitar. O caminho do hacker é o caminho do conhecimento, e esse conhecimento não se obtém em um curso, nem em vários. Ele vem por exaustivos anos de estudos teóricos e da aplicação dessas teorias na prática. Há inúmeros **lammers** (o texto original grafa assim; a forma mais comum é *lamer*) que falam o contrário. Muitos deles, inclusive, são *best sellers* na Udemy, amam o Kali GNU/Linux e passam o dia em fóruns discutindo com outros lammers. Reparo nos dados estatísticos dos meus conteúdos que o material básico e teórico é negligenciado pelos alunos, enquanto os conteúdos "hacker" são muito acessados — o que mostra o interesse imediatista de futuros lammers.

Mas qual é a função de um lammer no mundo hacker? Ele será entregue às autoridades e levará toda a culpa; os verdadeiros hackers farão isso no final da campanha. Guarde essa frase, porque ela volta várias vezes neste capítulo. Esta obra não o capacitará para ser um hacker completo, mas vou descrever aqui parte do meu conhecimento focado neste assunto — conhecimento obtido em anos de estudo de livros e anos de prática, tanto do lado da Luz quanto do lado mais obscuro do hacktivismo e do terrorismo digital.

O caminho é tortuoso e demorado. E em que você vai se formar no final? A resposta é simples: você será melhor do que era antes, e sempre terá alguém melhor do que você. Isso o leva ao ciclo infinito deste caminho eterno, pois sempre que evoluímos neste nível avançado acabamos nos viciando em evoluir e em ser melhores. Talvez a pergunta que poucos me fazem seja até onde posso ir nesse sentido. De novo, a resposta é simples: aonde conseguir ir. Você terá o domínio tecnológico adequado para atingir seus fins pessoais, e o teto é o seu próprio empenho.

Muitos de nós, hackers, nos contentamos em avançar em uma tecnologia, corrigir uma falha, ajudar uma empresa. Já outros chegam ao ponto de atacar organizações, países e até outros seres humanos. É como eu disse: seus fins pessoais, e eu não vou criticar. Este é um livro que descreve o território, não um púlpito.

A legalidade, e por que ela é uma linha borrada

Outra questão sobre a qual me questionam é a legalidade de tudo isso, ou o motivo pelo qual fazemos o que fazemos. Vou contar uma história breve. Em cursos de segurança, é clássica a pergunta de prova em alguma disciplina: **você contrataria um hacker sabendo que ele é um hacker ativo e de alta capacidade ofensiva contra ambientes?** Eu já fui questionado sobre isso em uma avaliação e respondi que SIM — fui o único, num universo de mais de 50 alunos. Lógico que "respondi errado". Mas eu me questiono sobre a legalidade de tudo: do

café na minha mesa que me vicia, da coca-cola que arrebenta meu fígado, de políticos que roubam, roubam e nada acontece, de montadoras que liberam carros no mercado com falha no cinto de segurança porque um *recall* sai mais barato.

Quem sabe ter um hacker muito bom trabalhando comigo, do meu lado, possa me ajudar a me defender de outros hackers tão bons quanto ele? Vejo o lado ético se isolando em um cenário ideológico perfeito, e vejo nisso uma grande falha de segurança organizacional — pois só um hacker vai entender outro hacker. E garanto: um dia me entenderão.

Vamos pegar o **Pegasus** (mais adiante vou lhe ensinar a programar um *spyware*), um software legítimo distribuído por vários governos para monitorar seus habitantes locais. Posso citar Israel, Estados Unidos, Inglaterra, França e outros. É um *malware*, e é legítimo e legalizado para e pelo Estado. O mesmo se diz do **Cobalt Strike**, que tem até sede própria, com programadores batendo cartão, e é um *malware*. Espero quebrar essa ideia de White e Black, de Certo e Errado, de Bem e Mal, pois é tudo um jogo definido nas nossas estruturas cognitivas. Por isso, no título deste trabalho, deixo claro: **ENTRE a Luz e as Trevas**. Não há dicotomia justamente por ser *entre*.

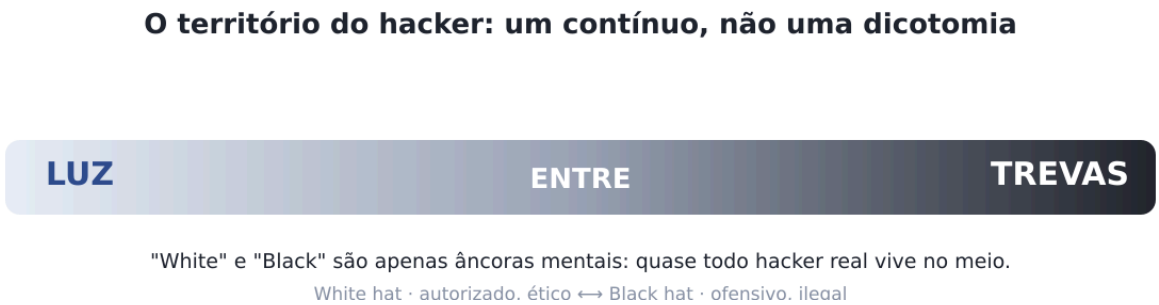

Figura: O território do hacker — um contínuo entre a Luz e as Trevas, não uma dicotomia.

Sobrevivência: o que nenhum livro te ensina primeiro

Espero estar claro sobre a legalidade do tema e a importância do conhecimento. Mas nenhum livro vai lhe dar uma teoria específica: **sobrevivência neste mundo hacker**. Vou lhe dizer que nem sempre publiquei conteúdos desse tipo. Sempre guardava para mim, e o foco dos meus textos era sempre GNU/Linux e Unix. Com o tempo, iniciei a publicação desses conteúdos de forma aleatória — há inúmeros vídeos que, embora apareçam depois, foram gravados antes dos iniciais. A construção deste material é e será caótica quanto à sequência.

Nesse processo, por várias vezes não fui anônimo no início. Recebi, até 2022, **duas ameaças de morte, inclusive uma presencial** — foram atrás de mim fisicamente. Ao expor meu conhecimento, cravei em mim um alvo. Recomendo que, de início, você foque nessa sobrevivência: na certeza de que não estará em perigo, para só depois focar em outras questões hacker. Foquei o começo deste livro nesse ponto para lhe ensinar o básico de sobrevivência que eu tive que aprender depois — mas saiba que é um assunto sempre em evolução. A regra que fica é dura: **o anonimato não é um recurso avançado que você liga quando fica bom; é a primeira coisa que você constrói, antes de tocar em qualquer alvo.**

Tipos de hackers

Eu não gosto de segmentar a área hacker, mas o mercado — e existe um grande mercado de livros e cursos que só formam fracassados — costuma segmentar "o mundo hacker" por temáticas. Vou descrever aqui as principais. Antes, porém, lembre-se de uma coisa que

atravessa todo este capítulo: **você não escolhe qual tipo quer ser; você acaba se encaixando naturalmente**, empurrado pelo seu talento, pela sua ética pessoal e pelas circunstâncias. As categorias a seguir não são gavetas fechadas — são regiões de um mesmo espectro, e a maioria dos hackers reais transita entre elas ao longo da vida.

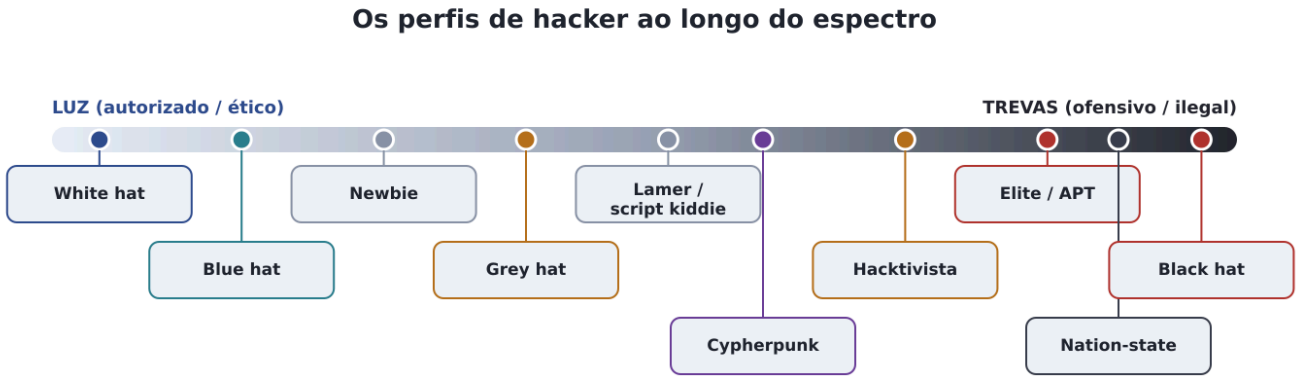

Figura: Os perfis de hacker distribuídos ao longo do espectro.

White hat

O **White hat** (chapéu branco) é o hacker que estuda sistemas de computação à procura de falhas de segurança, respeitando a ética hacker. Ao encontrar uma falha, ele normalmente se comunica em primeiro lugar com os responsáveis pelo sistema, para que tomem as medidas cabíveis. Muitos white hats desenvolvem suas pesquisas como professores universitários ou como empregados de empresas.

Eu acompanho inúmeros laboratórios que mantêm pesquisadores White Hat destrinchando ataques em nível global — e, pelo que conheço do mundo Black Hat, os coleguinhas de lá também fazem isso. Existem competições de busca de erros, os programas chamados **bug bounty**, em que as empresas expõem ambientes fechados, *labs* prontos. O White Hat que participa desses programas geralmente não ataca o ambiente de produção; quem faz isso é o Black Hat. Muitas empresas, com medo de deixar programas tão abertos, mantêm as próprias equipes de hackers éticos. Vejo nisso um erro grave de estratégia: um grupo fechado e pequeno sempre será mais limitado do que um grande grupo volátil, do tamanho do que se consegue num programa público de bug bounty.

Eu comecei nesse nível. Fui responsável, em 2017, por localizar uma falha grave de autenticação no sistema da empresa em que trabalhava. Prontamente notifiquei meus superiores, que corrigiram tudo em três dias. A falha estava em um componente terceiro chamado **Genexus**, e por isso impactava também os demais clientes da Genexus, incluindo Mitsubishi, Santander e outras grandes corporações. Mesmo assim, passei a ser discriminado por minhas ações por algumas pessoas. E fica o aviso técnico: caso você não esteja dentro da empresa que possui a falha, o correto é seguir o **VDP** (*Vulnerability Disclosure Program*), que descrevo no capítulo de Vulnerabilidade. **Notificar uma vulnerabilidade pode ser arriscado.** No mundo real, quem avisa a falha às vezes vira réu antes de virar herói.

Para pôr rosto nessa categoria, vale conhecer os nomes que a definem. **Charlie Miller** e **Chris Valasek** demonstraram, em 2015, o controle remoto de um Jeep Cherokee em movimento pela rede celular do carro — um trabalho que forçou um *recall* de 1,4 milhão de

veículos pela Fiat Chrysler e virou marco da segurança automotiva. **Katie Moussouris** criou o primeiro programa de bug bounty da Microsoft e foi peça central na formalização da divulgação coordenada de vulnerabilidades. **Tavis Ormandy**, do Google Project Zero, é conhecido por caçar falhas devastadoras em antivírus e em bibliotecas usadas por bilhões de pessoas. São white hats no sentido pleno: capacidade ofensiva de sobra, canalizada para o lado da correção.

Black hat

O **Black hat** (chapéu preto) é sinônimo de *cracker*: um hacker que não respeita a ética e usa seu conhecimento para fins criminosos, geralmente envolvendo ganho financeiro. Muitos hackers sonham em chegar a esse nível, e eu não os recrimino — acredito que a pessoa deve construir seu próprio caminho e ser responsável pelos seus atos. E é claro que vou lhe ensinar os macetes desse ambiente. **Eu não consegui me aceitar aqui, e por isso não estou aqui** — mas isso não me impede de descrevê-lo com honestidade.

Muitos falsários e estelionatários que não são hackers utilizam ferramentas ou serviços desses hackers Black. Saiba, então, que nem todo ataque é feito por um hacker. Os Black Hat que atacam empresas geralmente buscam uma ou duas situações a partir da vulnerabilidade encontrada:

1. **Extorsão para liberar dados da vulnerabilidade** — situação simples: com criptomoedas é possível fazer uma transferência segura e anônima (bem usada, ou com moedas voltadas ao anonimato como o **Monero**), e junto ao envio da falha o hacker costuma deixar uma *flag* no sistema, uma marca comprobatória de que entrou.
2. **Obtenção de acesso seguido de extorsão** — que, no geral, se desdobra em:

- **A. Sequestro do ambiente**, para fins diversos: ataques secundários a outros alvos, uso como ponte/proxy, armazenamento, e por aí vai — a "sobrevivência no planeta alienígena".

- **B. Exfiltração de dados** (chamamos de *exfiltrar*) para extorsão via LGPD, de preferência dados sensíveis.

- **C. Criptografia de dados** para extorsão.

Hoje os hackers aprenderam que podem usar a LGPD para forçar o pagamento. No caso 2.B, após uma exfiltração complexa, os grupos publicam parte dos dados em algum fórum, deixam bem público e usam **torrent** para difundir essa amostra — que pode variar de 5% a 50% do total, como venho observando. Segue-se a extorsão para não liberar o restante. Já no caso 2.C, os grupos entram pela vulnerabilidade e executam alguns *malwares*: tipicamente um **Worm** se alastra horizontalmente na rede levando consigo um **Ransomware** (vou lhe ensinar a programar tais artefatos no futuro). Com os dados criptografados e irrecuperáveis, começa a extorsão.

No passado, 2.B era irrisório e nenhuma empresa pagava — por isso o medo excessivo de 2.C, que se popularizou em 2017 e é hoje o maior pesadelo corporativo. Mas, graças à LGPD, os hackers passaram a usar o que chamamos de **dupla extorsão** (*double extortion*): unir 2.B e 2.C num único ataque, cobrando tanto para descriptografar quanto para não vazar. Como os hackers aprenderam a jogar a empresa contra a parede, vou contar um segredinho de como agem: primeiro se notifica a criptografia (2.C) e se aguarda algumas horas. Geralmente as empresas usam *backups* como plano de continuidade, o que acho um erro grave (recomendo a leitura do **NIST 800-34** para entender esse assunto). Imagine a diretoria passando horas em reunião decidindo se paga, enquanto pressiona a TI para executar o Plano de Continuidade de Negócio. Então, quando estiverem bem nervosos e estressados, publica-se 5% dos dados e se

notifica a publicação de mais 5% a cada 8 horas. A decisão fica abalada, e a TI, exausta, não consegue reagir. Descrevo o mecanismo não para aplaudir, mas para você entender o inimigo — a defesa começa em saber como o ataque pensa.

Se você quiser criar artefatos para outros hackers, essa atividade se enquadra neste segmento: um Black Hat pode comprar tais componentes para os ataques. A venda desses artefatos é feita em vários fóruns hacker, principalmente na **Deep Web** (de 100 a 25.000 dólares — geralmente mais caro no lançamento, caindo de preço com o tempo), e pode ser monitorada por projetos como o VulDB (`https://vuldb.com/pt/`). Digamos que é lucrativo.

Para dar nome aos bois, o ecossistema Black Hat moderno é dominado por grupos de *ransomware* organizados como empresas. **LockBit** operou por anos o modelo de *Ransomware-as-a-Service*, alugando seu malware a afiliados. **Conti** vazou seus próprios chats internos em 2022, revelando uma estrutura corporativa com RH, folha de pagamento e metas — antes de se fragmentar. **REvil** (também chamado Sodinokibi) ficou famoso pelos ataques à Kaseya e à JBS. **BlackCat/ALPHV** foi um dos primeiros grupos de peso a escrever seu ransomware em **Rust**. E o **Clop** especializou-se em explorar falhas de ferramentas de transferência de arquivos (como MOVEit e GoAnywhere) para exfiltrar dados de centenas de organizações de uma vez. Todos eles operam exatamente a dupla extorsão descrita acima, com "sites de vazamento" na rede Onion onde publicam as amostras.

Grey hat

O **Grey hat** (chapéu cinza) é o hacker intermediário entre White Hat e Black Hat. Por exemplo: um Grey Hat invade sistemas por diversão, mas evita causar danos sérios e não copia dados confidenciais. Muitos desses hackers têm o sonho de conseguir um grande emprego e tentam obtê-lo basicamente invadindo empresas — o que não sabem é que menos de 0,00001% vão conseguir, e a imensa maioria será vista como criminosa. Existe um hacker conhecido como Commander, que invadia assim, até precisar de dinheiro e migrar para o submundo Black Hat. No meu ponto de vista, esse é um cenário transitório: ninguém fica cinza para sempre — a pressão da vida empurra para um dos lados.

Elite hacker

O **Elite hacker** é uma reverência dada apenas aos hackers exímios, e constitui um elevado *status* dentro da comunidade hacker ou dentro de um **APT** (*Advanced Persistent Threat*, os grupos hackers persistentes e avançados). Os grupos APT são, em suma, hierárquicos, com vários níveis. No topo da hierarquia está o Elite Hacker coordenando as campanhas. É raro que um deles ponha a mão diretamente na operação; são sempre protegidos e, em geral, não são assassinados nas purgas constantes que acontecem dentro dos grupos APT. A elite pensa a campanha; a base a executa — e a base é descartável.

O termo "APT" foi cunhado justamente para descrever atacantes que não fazem um golpe e somem, mas que se instalam e persistem por meses ou anos. Os exemplos reais mais estudados são estatais. A **APT28**, apelidada de **Fancy Bear**, e a **APT29**, a **Cozy Bear**, são atribuídas à inteligência russa e ficaram conhecidas por operações de espionagem e interferência política. O **Lazarus Group**, ligado à Coreia do Norte, mistura espionagem com roubo financeiro puro — do ataque à Sony Pictures ao furto de bilhões em criptomoedas. A **APT1**, exposta em 2013 pelo relatório da **Mandiant**, foi identificada como a **Unidade 61398** do Exército chinês, dedicada a roubo de propriedade intelectual em escala industrial. E o **Equation Group**, revelado pela Kaspersky, é amplamente associado à NSA e considerado

tecnicamente o mais sofisticado já documentado. Estudar esses grupos é estudar a elite — não a lenda de fórum, mas a engenharia real por trás das campanhas.

Newbie

Muitas vezes abreviado como "NB", **Newbie** é o termo usado, em sentido pejorativo, para designar um hacker principiante. Em grupos hacker, pode servir para depreciar outro membro, e é bem chato de ouvir. Mas atenção: não quer dizer que o hacker seja um Lammer. Ele é apenas um principiante, e deve engolir o termo para poder estar naquele ambiente. Todo mundo começa aqui — a diferença entre o Newbie e o Lammer não é o quanto sabe hoje, é a disposição de estudar amanhã.

Blue hat

O **Blue hat** (chapéu azul) é um hacker contratado por empresas para encontrar vulnerabilidades em seus sistemas **antes** dos lançamentos. Quando uma equipe de desenvolvimento atua, ela é pressionada para entregar requisitos funcionais — e na área de TI temos uma vida ingrata quanto à parte laboral. Nessa pressão, e na falta de entendimento entre **DEV** e **OPS**, inúmeras vulnerabilidades passam para produção (a sua empresa deveria seguir **DevSecOps**; vai dar problema se não adotar tais práticas). Mas lembre-se: nem toda vulnerabilidade é falha de desenvolvimento ou de operação — a negação de serviço, por exemplo, não é (um Lammer não entenderia). O Blue Hat pode atuar agregado ao White Hat em todo o processo, entrando cedo, ainda no pré-lançamento.

Lamer

O **Lamer** — ou **script kiddie** ("moleque de script") — é alguém que se considera hacker, mas tem preguiça de estudar ou de evoluir. Vira então um "executador" (não existe essa palavra) de scripts, rotinas ou programas, sem saber o motivo do que faz, porque fez um CURSO HACKER escrito por outro Lammer que só quer vender curso. Geralmente o Lamer deixa rastros: por não ter conhecimento, faz lambanças na entrada e na saída de um ambiente alvo e é facilmente detectado. A falta de conhecimento o torna pouco efetivo. Um hacker experiente, dotado de conhecimento, executa uma ação certeira, de tal forma que se torna indetectável. Em grupos hacker, esses Lammers são enganados e entregues aos Leões no final da campanha — pagam pela preguiça de estudar os livros teóricos densos e cheios de conceitos. Repito a frase-chave deste capítulo porque ela é literal: **o Lamer é a carne que o grupo deixa para a polícia levar.**

Phreaker

O **Phreaker** é um hacker especializado em telefonia, seja móvel ou fixa. E, por incrível que pareça, é o primórdio de todo o tema hacker. Antes de existirem computadores pessoais em rede, existiam as centrais telefônicas — e foi ali que nasceu a cultura de explorar sistemas por curiosidade e pela beleza do próprio truque.

A história canônica do *phreaking* é a do **John Draper**, apelidado de **Capitão Crunch** (*Captain Crunch*): ele descobriu que o apito de brinde de uma caixa de cereal emitia exatamente o tom de **2600 Hz** que sinalizava linha livre nas centrais da AT&T, permitindo fazer chamadas de longa distância gratuitas. Dessa descoberta nasceram as **blue boxes**, aparelhos que geravam esses tons — e dois jovens chamados **Steve Wozniak** e **Steve Jobs** montavam e vendiam blue boxes antes de fundarem a Apple. Aquele número virou nome de cultura: a revista **2600: The Hacker Quarterly**, referência do movimento até hoje,

homenageia justamente a frequência de 2600 Hz. Quem entende phreaking entende que "hackear" nunca foi sobre computador — foi sempre sobre entender o sistema melhor do que quem o construiu.

Hacktivista

O **Hacktivista** (*hacktivist*) é o hacker que se envolve em ciberterrorismo — não gosto desse termo, talvez por me encaixar em parte dele no meu dia a dia — ou que usa suas habilidades em favor de causas sociais, políticas, ideológicas ou religiosas. Muitos grupos se encaixam no termo "anônimos" e atuam nesse submundo. Curiosamente, os "anônimos" começaram em fóruns baseados em tópicos, fóruns que não exigiam cadastro: o programador desses fóruns simplesmente usou a palavra **Anonymous** porque precisava exibir alguma coisa no campo de usuário. Foi uma decisão de programador que criou um termo mundialmente conhecido.

A diferença desses milhares de grupos descentralizados que se intitulam Anonymous, se comparados aos APTs, é o anonimato total e a descentralização. Por isso é possível ver "anonymous" atacando tanto governos de direita quanto de esquerda em diferentes países (embora, no meu ponto de vista, boa parte dos que se dizem anonymous seja de revoltados com o mundo capitalista). No campo dos exemplos reais: o coletivo **Anonymous** consolidou a estética das máscaras de Guy Fawkes e das operações "#Op"; o **LulzSec**, um grupo menor e mais afiado que se desprendeu do Anonymous em 2011, fez uma sequência barulhenta de invasões (Sony, órgãos de segurança) "pelas risadas" e implodiu quando seu líder, o **Sabu**, foi preso e colaborou com o FBI — um exemplo perfeito da frase sobre quem é entregue no final. Fora do eixo anglófono, grupos como o turco **RedHack** mostram como o hacktivismo se enraíza em pautas locais.

No meu ponto de vista, esse é o estágio final — se você não for pego pelo governo. Ao chegar a esse ponto, com tanto conhecimento e tão crente nas próprias convicções, a destruição da sua vida e da sua carreira se torna quase inevitável, em nome de algum ataque ou ação em favor de uma causa. O hacktivismo é o lugar onde a técnica encontra a fé — e a fé cega para as consequências.

Cypherpunk

O **Cypherpunk** é, antes de tudo, um construtor. Geralmente detém muito conhecimento de desenvolvimento e, principalmente, de **criptografia**. Sobreviver em ambientes onde o próprio Estado é o criminoso exige ferramentas, e essas ferramentas garantem o anonimato — entenda anonimato aqui como um **agorismo**, uma forma de agir livre à margem do controle estatal. Ferramentas complexas são desenvolvidas por esses elementos, e por isso afirmo: são esses hackers que constroem as bases da nossa sociedade hacker. São exemplos o **Whonix** GNU/Linux, o **Kodachi** GNU/Linux, o **Tor**, as criptomoedas, ferramentas de cifragem, e por aí vai. Nos últimos anos venho me aprimorando e adentrando esse mundo, e posso dizer: é fantástico. **Eu me encontrei aqui.**

O movimento tem berço e documento fundador. O **Manifesto Cypherpunk** (*A Cypherpunk's Manifesto*), escrito por **Eric Hughes** em 1993, defende a ideia de que "os cypherpunks escrevem código" — porque não adianta esperar que governos ou empresas nos deem privacidade; ela precisa ser construída por quem a valoriza. Ao lado de Hughes estavam nomes como **Timothy May**, autor do "Manifesto Cripto-Anarquista" (*The Crypto Anarchist Manifesto*), e **John Gilmore**, cofundador da Electronic Frontier Foundation. Dessa cultura saíram criações concretas: **Phil Zimmermann** escreveu o **PGP** (*Pretty Good Privacy*) e foi investigado pelo governo americano por "exportar munição" — porque criptografia forte era

classificada assim. E **Julian Assange**, antes do WikiLeaks, já era parte dessa lista de e-mails. Estudar os cypherpunks é entender de onde vieram todas as ferramentas de anonimato que este livro ensina a usar.

Nation-state professionals

Os **Nation-state professionals** são hackers (contratados, obrigados ou escravizados) por/de governos e agências de inteligência estatais, além de unidades específicas de guerra cibernética. Com enorme poderio computacional à disposição, eles visam:

- Atuar na **cyberwarfare** (guerra cibernética):

- atacando virtualmente serviços;

- atacando setores financeiros;

- atacando infraestruturas militares e civis;

- espionando;

- mudando rumos de eleições e alterando a política;

- Perseguir outros hackers.

Não acredito que trabalhar para governos seja um mundo lindo. É um ambiente extremamente político, e estar realizando práticas ilícitas para políticos pode ser péssimo: **quem brinca com fogo pode se queimar.** Os nomes reais desse mundo se confundem com os APTs já citados, mas convém conhecer as unidades por trás. A **TAO** (*Tailored Access Operations*) da NSA americana é a divisão de intrusão de elite revelada em vazamentos. A **Unidade 8200** de Israel é famosa por formar boa parte da indústria mundial de cibersegurança a partir de seus veteranos. A chinesa **Unidade 61398** é a face concreta da APT1 do relatório Mandiant. E a **GRU** (a inteligência militar russa) opera por trás da Fancy Bear. São esses os "profissionais" — com carteira assinada por Estados, imunidade dentro de suas fronteiras e alvo pintado nas costas fora delas.

Aonde me encaixo?

Eu não quero ser um tipo. Na verdade, eu me encaixo em algum perfil — não necessariamente em um só. Não tenho estômago para roubar, então não sou Black; mas também acho os White um bando de idiotas que nunca vão me entender, nem entender um Black. O que eu acredito é que governos são monstros, e isso ocorre porque, por muitos séculos, nós cedemos um pouco de nós ao governo: um pouco da liberdade, dos bens, da vida.

Agora a luta é pesada, e às vezes parece uma eterna luta contra moinhos de vento. Mas aviso: o número de elementos no nosso exército é cada vez maior, e temos ferramentas para bater de igual para igual com governos — porque, no campo cyber, nós ditamos as regras. Aqui, nós somos a constituição. Por ter programado durante mais de 20 anos, é natural que essa visão anarquista fosse ao encontro do mundo da criptografia. Minha movimentação pelo mundo do cypherpunk é constante — e, se você prestou atenção às categorias acima, vai perceber que o objetivo delas nunca foi lhe dar uma etiqueta, e sim um mapa para reconhecer para onde os seus próprios esforços estão levando você.

Recomendações para início

Você não vai escolher o que vai ser: naturalmente, os seus esforços vão lhe gabaritar e encaixar em algum desses tipos. Se não quer estudar e quer só comprar cursos, vai virar um Lammer naturalmente. Se estudar, ficar fera em tecnologia e passar anos em grupos hacker agindo, poderá se encaixar como um hacker de elite e será reconhecido. As minhas recomendações concretas de estudo são estas:

- **Comprar cursos pode ajudar?** Pode, mas os cursos de hoje são focados em ferramentas e formam Lammers. Use-os como complemento, nunca como base.
- **Disciplinas para formar sua base:**

- Sistemas Operacionais (teoria) — no livro do **Tanenbaum**;

- Redes de Computadores (teoria) — também no **Tanenbaum**;

- Sistemas Operacionais na prática, com GNU/Linux;

- Serviços de rede, com GNU/Linux e Unix;

- Lógica de programação;

- Compiladores.

- **Linguagens e scripts para dominar com maestria:**

- **Python**;

- **C/C++**;

- **GoLang**;

- **Pascal/Delphi**;

- **C#** e muito **.NET Framework**;

- **VBScript**;

- **VBA** no Office;

- **PowerShell** (Windows);

- Script **CMD** (Windows);

- Script **Shell** (GNU/Linux).

Se você acha que é muito, então eu espero que nem avance neste livro — pode parar por aqui, porque você não tem aptidão para a nossa área. Continuar só vai perder o seu tempo e encher os fóruns com perguntas idiotas. Recomendo ir trabalhar com paisagismo. Digo isso sem ironia e sem raiva: este caminho consome anos, expõe você a riscos reais (como as duas ameaças de morte que descrevi) e não entrega nenhum diploma no fim. Se a lista acima te assusta em vez de te animar, o filtro já cumpriu a função dele. Se te animou — bem-vindo. O próximo passo é aprender a não morrer no processo, e é disso que tratam os capítulos seguintes.

Ferramentas citadas

- **Kali GNU/Linux** — distribuição Linux voltada a testes de intrusão; **código aberto** (GPL, mantida pela Offensive Security).
- **Pegasus** — *spyware* de vigilância para dispositivos móveis, desenvolvido pela NSO Group; **comercial** (venda restrita a governos).

- **Cobalt Strike** — plataforma de simulação de adversário e *post-exploitation*, muito abusada por Black Hats; **comercial** (Fortra).
- **Tor** — rede de roteamento anônimo (roteamento Onion); **código aberto** (licença BSD de 3 cláusulas).
- **Whonix** — sistema operacional voltado ao anonimato, que força todo o tráfego pelo Tor; **código aberto** (licenças livres/GPL).
- **Kodachi** — distribuição GNU/Linux focada em privacidade e anonimato; **código aberto**.
- **Monero (XMR)** — criptomoeda com foco em privacidade e anonimato de transações; **código aberto**.
- **PGP / GnuPG** — cifragem e assinatura de dados (o PGP original de Phil Zimmermann; o GnuPG é a implementação livre); **código aberto** (GnuPG sob GPL).
- **VulDB** — base de dados de vulnerabilidades e monitoramento de mercado; **comercial/freemium**.

Referências

- HUGHES, Eric. **A Cypherpunk's Manifesto**. 1993.
- MAY, Timothy C. **The Crypto Anarchist Manifesto**. 1988/1992.
- LEVY, Steven. **Hackers: Heroes of the Computer Revolution**. Anchor Press/Doubleday, 1984.
- MITNICK, Kevin; SIMON, William L. **A Arte de Enganar** (*The Art of Deception*). Wiley, 2002.
- MANDIANT. **APT1: Exposing One of China's Cyber Espionage Units**. 2013.
- TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. Pearson.
- TANENBAUM, Andrew S.; WETHERALL, David J. **Redes de Computadores**. Pearson.
- NIST. **SP 800-34: Contingency Planning Guide for Federal Information Systems**.
- BRASIL. **Lei nº 12.737, de 30 de novembro de 2012** (Lei Carolina Dieckmann — tipificação de crimes informáticos).
- BRASIL. **Lei nº 13.709, de 14 de agosto de 2018** (Lei Geral de Proteção de Dados — LGPD).

Identidade Digital e Autenticação

Seria doido se eu dissesse que você tem que ser doido, de ter várias personalidades? Não fique perplexo — sim, nós, hackers, devemos ter muitas personalidades frente à tecnologia, e isso vai dificultar a nossa localização. Antes de qualquer ferramenta, antes do Tor ou de uma máquina virtual, é preciso entender uma coisa: no mundo digital você não é uma pessoa, você é um conjunto de identidades, e a arte está em não deixar que esse conjunto se colapse em um único ponto que aponte para o seu corpo real.

A identidade digital é a representação única de uma pessoa real envolvida em uma experiência sobre algum recurso tecnológico. Para a Interação Humano-Computador, a experiência do usuário é o conjunto de ações desse sujeito sobre um recurso tecnológico — e, em muitas situações, a própria experiência do usuário já é a sua representação única. O jeito de digitar, os horários em que aparece, a sequência de cliques, o navegador que carrega: tudo isso desenha um contorno. Esse contorno tem nome técnico, **impressão digital de navegador** (*browser fingerprint*), e voltaremos a ele porque é justamente o que trai o anônimo.

Nesse cenário é possível compreender a relação entre um ser humano e a sua identidade virtual pela experiência que ele tem frente à tecnologia (o *fingerprint*). Neste livro serão abordadas as técnicas para reduzir a assertividade dessa relação e garantir ao máximo o anonimato. A ideia central não é ficar invisível — isso é quase impossível —, é embaralhar a ligação entre "o que se observa na rede" e "quem é a pessoa por trás".

A identidade digital é sempre única no contexto do recurso digital em que o usuário está tendo a sua experiência; mas, em outros contextos ou outros recursos, o mesmo indivíduo pode possuir outras identidades digitais. Alguns sistemas exigem alguma prova de ligação entre a identidade virtual do usuário e o ser humano real, mas nenhum desses métodos é infalível — o que, aliás, é um bom argumento em tribunais. Em outras palavras: acessar um serviço digital pode não significar que a identidade do ser humano na vida real seja conhecida. Esse "pode não significar" é a brecha inteira sobre a qual este capítulo se apoia.

Neste livro será construída uma identidade virtual que representa o autor com base em dados públicos, e também uma identidade fictícia de um hacker fictício — afinal, ao contrário da identidade física, a identidade virtual não é necessariamente única. Com o próprio material deste livro já se pode construir parte da informação sobre a identidade do autor. Um especialista da máfia governamental traçaria um perfil pelo que eles chamam de **metadados**. Vejamos um exemplo de metadados de um sujeito qualquer e o que se sabe atualmente sobre ele.

```
INDIVÍDUO cbeaf1d5-4b76-41be-ab0d-cbf3f285b232 = {
    "name": "Manoel Marcos Silva e Bueno",
    "actions_internet": [ "escreve", "grava vídeos" ],
    "knowledge": [ "informática" ],
    "register": [ "google" ]
}
```

Veja: se este livro está sendo editado em uma ferramenta na nuvem (*Cloud*), é natural que o autor tenha uma conta nessa nuvem. Será que se pode extrair alguma informação sobre esse usuário a partir da plataforma? A empresa entregaria a localização dessa pessoa? Respeitaria a **GDPR** (*General Data Protection Regulation*, o regulamento europeu) ou a **LGPD** (Lei Geral

de Proteção de Dados, a brasileira)? A pergunta não é retórica: essas leis definem quando um provedor pode — ou é obrigado a — reter e entregar seus dados, e nenhum hacker deveria depositar confiança em uma cláusula de privacidade que uma ordem judicial derruba em minutos.

Algo interessante é que esse ser tem um conhecimento que o possibilita transmitir conhecimento, logo é capaz de influenciar outras pessoas — e veja o lado bom da influência. Então é possível que ele esteja em redes sociais difundindo conhecimento somado a ideologias. Repare como um único atributo (`"knowledge": ["informática"]`) abriu uma inferência inteira sobre comportamento provável. É assim que o perfilamento funciona: não a partir de um fato, mas a partir do encadeamento de probabilidades.

Logo, logo esse perfil sobre a pessoa será tão grande que precisará de um banco de dados, porque há muitíssima informação pública sobre as pessoas na Internet. Repare que os dados **não são estruturados** — ou seja, exigem grandes bancos de dados documentais e distribuídos, e os mecanismos de processamento não podem ser desenvolvidos com a lógica clássica. O correto é trabalhar com **lógica difusa** (*fuzzy logic*). A lógica clássica responde "sim" ou "não"; a lógica difusa responde com um grau de pertinência entre 0 e 1 — "esse perfil é 0,82 provavelmente a mesma pessoa daquele outro". É exatamente esse tipo de raciocínio que motores de correlação usam para amarrar contas que você achou que estavam separadas.

Metadados: a sombra que fala por você

Vale insistir no ponto porque ele sustenta tudo o que vem depois. Metadado é o dado sobre o dado: não é o conteúdo da sua mensagem, é o fato de que houve uma mensagem, a que horas, de qual dispositivo, para quem. O conteúdo você pode cifrar; o metadado quase sempre vaza. Um exemplo mais completo do mesmo indivíduo, já enriquecido por um agregador, se pareceria com isto:

```
INDIVÍDUO cbeaf1d5-4b76-41be-ab0d-cbf3f285b232 = {
    "name": "Manoel Marcos Silva e Bueno",
    "actions_internet": [ "escreve", "grava vídeos", "publica de madrugada" ],
    "knowledge": [ "informática", "segurança" ],
    "register": [ "google", "cloud_editor", "rede_social_X" ],
    "timezone_inferido": "UTC-3",
    "dispositivos": [ "android", "notebook_linux" ],
    "confianca": 0.82
}
```

Nenhum desses campos foi confessado pelo alvo. Todos foram inferidos: o fuso horário saiu dos horários de postagem, o sistema operacional saiu dos cabeçalhos das requisições, o campo `confianca` é a saída de um modelo difuso que amarra registros dispersos. É por isso que a defesa do hacker começa por um princípio antes de qualquer ferramenta: a **minimização de dados** (*data minimization*), princípio expresso na própria LGPD e na GDPR — não forneça, não gere e não deixe correlacionável nenhum dado além do estritamente necessário para a operação em curso. Cada campo a mais nesse JSON é um degrau a mais para quem te caça.

Autenticação Digital

A autenticação digital é o processo de determinar a validade de um ou mais meios de autenticação usados para reivindicar uma identidade e, naturalmente, uma unicidade em meio à população (não necessariamente real). É o processo preferido para permitir o acesso a certos recursos. Veja: se um usuário recebe um e-mail, é natural que se exija dele a prova de que é o dono da mensagem e pode lê-la. Autenticar é responder à pergunta "você é mesmo quem diz ser?" — e cada resposta a essa pergunta é uma oportunidade de amarrar o digital ao real.

Esse mecanismo, quando obtém êxito, garante parcialmente a lisura do processo e garante que a pessoa dona de uma identidade digital possa ser localizada no mundo real: é uma **chave estrangeira** (*foreign key*) entre o digital e o real. Esse é o desafio. Do lado de quem defende o sistema, essa chave estrangeira tem um nome de mercado — **KYC** (*Know Your Customer*, "conheça o seu cliente") —, e em contexto financeiro vem acompanhada do **AML** (*Anti-Money Laundering*, prevenção à lavagem de dinheiro). São exatamente os regimes que obrigam bancos, corretoras e cada vez mais plataformas comuns a exigir documento, selfie e prova de vida antes de liberar uma conta. Para o hacker, KYC é a muralha que transforma uma conta anônima numa conta rastreável; entender onde ele é obrigatório e onde não é já é meia operação. Vamos imaginar alguns cenários.

O **Cenário 1** é o roubo de documentos e de celular. Sim — o ser humano se preocupa mais com os documentos e os cartões de crédito, e pouco pensa no número de celular roubado. Isso é um erro. A perda de uma simples linha telefônica é catastrófica: um hacker pode criar inúmeras contas em inúmeros sistemas e atrapalhar a vida do dono real, e os prejuízos podem ser até maiores do que o saldo dele no banco. O motivo é que a linha telefônica virou a chave-mestra da recuperação de senha: quem controla o número recebe os códigos de "esqueci minha senha" de e-mail, banco e redes sociais, e a partir daí toma tudo em cascata.

O **Cenário 2** é o roubo dos *cookies* de um navegador. Algo tão simples pode ser danoso: aplicações modernas utilizam demasiadamente os *cookies* como armazenamento de dados sobre o usuário, e esses dados podem ser usados em ataques posteriores. O caso mais grave é o **cookie de sessão**: ele representa uma sessão *já autenticada*. Quem o rouba não precisa da sua senha nem do seu segundo fator — apenas cola o *cookie* no próprio navegador e entra como você, porque o servidor não vê a senha, vê o *token* de sessão válido. É o motivo pelo qual roubar sessão hoje vale mais que roubar senha.

A identidade digital apresenta um desafio técnico porque esse processo geralmente envolve a verificação de indivíduos em uma **rede aberta** e, normalmente, envolve a autenticação de indivíduos em uma rede aberta para acessar serviços digitais importantes para a vida dessas pessoas. Existem várias oportunidades de roubo de identidade e outros ataques que reivindicam de forma fraudulenta a identidade digital de outro sujeito. "Rede aberta" aqui é o ponto: você prova quem é atravessando uma infraestrutura que não controla e onde qualquer intermediário pode observar, interceptar ou personificar.

A autenticação digital oferece suporte à proteção da privacidade, reduzindo os riscos de acesso não autorizado às informações dos indivíduos. Ao mesmo tempo, como a prova de identidade, a autenticação e a autorização envolvem o processamento de informações de indivíduos, essas funções também podem criar riscos de privacidade. Por isso, diretrizes sérias sobre o tema — como a norma norte-americana **NIST SP 800-63-3**, referência mundial em identidade digital — incluem requisitos de privacidade e considerações para mitigar potenciais riscos. Essa norma, aliás, é útil ao hacker por um motivo inesperado: ela descreve com precisão os **níveis de garantia de autenticação** (*Authenticator Assurance Levels*) que um sistema pode exigir, e saber em que nível um serviço opera é saber o quanto de esforço

ele fará para ligar sua conta ao seu corpo. Para manter uma segurança devida, deve-se utilizar:

- Um segundo fator de autenticação **de verdade** (por exemplo, uma chave de hardware **Yubico / YubiKey**);
- **OAuth**, para não espalhar credenciais por todo lado.

Convém ser honesto sobre o que é "de verdade" aqui, porque a indústria vendeu como segundo fator várias coisas que não são. O **2FA** (autenticação de dois fatores) só cumpre o que promete quando o segundo fator resiste a *phishing*. Vamos por camadas:

- **SMS como segundo fator** é o pior de todos: o código viaja pela rede da operadora, pode ser interceptado, redirecionado ou obtido por engenharia social junto à operadora. É a porta do *SIM swapping*, que veremos adiante.
- **Aplicativo autenticador** (*TOTP*, os códigos de seis dígitos que giram a cada trinta segundos) é bem melhor que SMS, porque o segredo fica no seu dispositivo e não trafega — mas ainda pode ser *phishado*, pois nada impede você de digitar o código num site falso.
- **Chave de hardware e passkeys** são o padrão-ouro. Aqui entram o **FIDO2** e o **WebAuthn** (a especificação do W3C que os navegadores implementam), promovidos pela **FIDO Alliance**. O segredo é uma chave criptográfica que **nunca sai** do dispositivo (a YubiKey, o telefone, o chip do computador), e a assinatura é atrelada ao domínio real do site. Se você cair num site de *phishing*, a chave simplesmente se recusa a assinar, porque o domínio não bate. As **passkeys** são a forma amigável dessa mesma tecnologia, sincronizadas entre dispositivos, e representam o caminho para o fim da senha. É a isso que o autor se refere ao dizer "segundo fator de verdade": um fator que a vítima não consegue, nem querendo, entregar a um golpista.

Sobre o **OAuth**: ele é um protocolo de **autorização** (não de autenticação, embora o **OpenID Connect** construa autenticação por cima dele). A ideia é você autorizar um aplicativo a agir em seu nome sem lhe entregar a sua senha — o app recebe um *token* de acesso limitado, revogável, com escopo definido. Do ponto de vista defensivo, isso reduz a superfície: sua senha fica em um único lugar. Do ponto de vista do hacker, é uma faca de dois gumes — o "entrar com o Google" é comodíssimo, mas amarra todas as suas contas satélites a um único provedor que sabe exatamente onde você entra e quando, um belo ponto único de correlação a evitar quando o objetivo é dispersão de identidade.

Criando personas: os erros que denunciam o operador

Quando um hacker cria uma personalidade virtual falsa, deve estar atento a algumas observações. A primeira diz respeito à conta (*account*): não se pode criar duas ou mais contas falsas com nomes semelhantes achando que se está imune ao algoritmo. Exemplo: **Kim Jong Um** e **Kim Jong Dois** — qual é a chance de ambos serem a mesma pessoa? O primeiro filtro agrupa esses dois usuários, e a partir daí a análise fica mais fácil. O erro parece bobo escrito assim, mas é exatamente o tipo de padrão mental (o trocadilho, a sequência, a piada interna) que um motor de correlação adora, porque revela uma única cabeça por trás de várias contas.

Embora seja idiota ter de dizer: a senha das contas falsas nunca pode ser igual — e também não pode ser semelhante — às das contas reais, visto que usuários podem ser relacionados pela senha. Isso não é teoria. Vazamentos de bases inteiras de senhas são cruzados o tempo todo, e uma senha reutilizada (ou uma variação óbvia dela, `MinhaSenha1`, `MinhaSenha2`) costura persona falsa a identidade real num só passo. A regra é dura: cada persona é um universo isolado — senha própria, e-mail próprio, número próprio, dispositivo ou perfil de

navegador próprio, e de preferência horários e estilo de escrita distintos. Qualquer campo compartilhado é uma solda entre dois mundos que deveriam permanecer separados.

O problema do SMS: ativação, IMEI e SIM swapping

Mas o mais difícil para o hacker é passar pelo processo de ativação de serviços por SMS — e isso o leva a um problema. Mesmo que o SMS não registre o **IMEI** (o número de série único do aparelho), uma incursão ditatorial da máfia estatal pode obter esses dados junto à operadora; e ainda é possível localizar a posição de um equipamento com certa precisão. Mas a precisão não importa: o que importa é que as pessoas daquela região passarão a fazer parte da investigação, e tais máfias possuem recursos ilimitados. A ativação de celular será discutida adiante no livro, mas o princípio fica: o número de telefone é o cordão umbilical mais curto entre a persona e o corpo.

Aqui é preciso nomear com precisão o ataque que assombra o Cenário 1 e a ativação por SMS: o **SIM swapping** (ou *SIM swap*, a "troca de chip"). O golpista convence a operadora — por engenharia social, suborno de um funcionário, ou com documentos vazados — a portar o **seu** número para um chip **dele**. No instante em que a portabilidade se conclui, o seu celular perde sinal e todos os seus SMS de recuperação e todos os seus códigos 2FA por mensagem passam a chegar no aparelho do atacante. A partir daí, ele dispara "esqueci minha senha" em e-mail, banco e redes sociais e coleciona os códigos. Foi assim que se esvaziaram carteiras de criptomoedas e se sequestraram perfis de altíssimo valor. A lição tem duas faces: como defensor, tire o SMS do caminho crítico e use FIDO2; como atacante consciente, entenda que **qualquer** conta sua ancorada num número de telefone herda a fragilidade da operadora, que é humana e subornável.

Mesmo depois de resolvido o problema da ativação — como veremos —, cada conta falsa terá alguns requisitos. Separo dois métodos de se obter um SMS:

- Um celular novo, um chip novo e dados fictícios;
- Comprar um SMS virtual na Internet.

Cada método tem o seu preço e o seu rastro. O celular e o chip novos custam dinheiro e exigem cuidado físico (onde foi comprado, com qual cartão, sob qual câmera, ligado em qual antena), mas te dão controle total do número. O SMS virtual comprado na Internet é barato e descartável, porém você não controla o serviço do outro lado: o número pode ser reciclado, o intermediário guarda logs e pode ser comprometido, e muitos serviços já reconhecem e bloqueiam as faixas de números virtuais mais conhecidas. A escolha entre um e outro é sempre um cálculo entre custo, descartabilidade e a quantidade de rastro que se pode tolerar — o mesmo cálculo que percorre este livro inteiro.

Note que os dois cenários de roubo, os métodos de ativação e a criação de personas não são assuntos separados: são o mesmo diagrama visto de ângulos diferentes. Existe um ciclo — o **ciclo de vida da identidade digital** — que vai da criação de uma persona, passa pela ligação a um fator de autenticação (número, e-mail, chave), atravessa o uso cotidiano que gera metadados, e termina na correlação que pode colapsar tudo de volta ao corpo real. Dominar o capítulo é dominar cada estágio desse ciclo antes que o adversário o feche por você.

Ferramentas citadas

- **YubiKey (Yubico)** — chave de segurança física que implementa FIDO2/WebAuthn, TOTP e outros protocolos de segundo fator resistente a *phishing*; **hardware**

comercial, com bibliotecas cliente de código aberto.
- **OAuth 2.0 / OpenID Connect** — protocolos abertos de autorização e autenticação delegada; **padrão aberto** (IETF / OpenID Foundation), implementações livres e comerciais.
- **FIDO2 / WebAuthn** — padrão aberto de autenticação sem senha e segundo fator por chave criptográfica; mantido pela **FIDO Alliance** e pelo **W3C**; **especificação aberta**, com implementações de código aberto e comerciais.

Referências

- NIST. **SP 800-63-3: Digital Identity Guidelines** (e volumes 63A/63B/63C). National Institute of Standards and Technology, 2017. Disponível em: https://pages.nist.gov/800-63-3/
- BRASIL. **Lei nº 13.709, de 14 de agosto de 2018 — Lei Geral de Proteção de Dados Pessoais (LGPD)**. Brasília, 2018.
- UNIÃO EUROPEIA. **Regulamento (UE) 2016/679 — Regulamento Geral sobre a Proteção de Dados (GDPR)**. 2016.
- FIDO ALLIANCE. **FIDO2: WebAuthn & CTAP — Specifications Overview**. Disponível em: https://fidoalliance.org/fido2/
- W3C. **Web Authentication: An API for accessing Public Key Credentials (WebAuthn) — Level 2**. World Wide Web Consortium, 2021.
- HARDT, Dick (ed.). **RFC 6749: The OAuth 2.0 Authorization Framework**. IETF, 2012.
- ZADEH, Lotfi A. **Fuzzy Sets**. Information and Control, v. 8, n. 3, 1965. (fundamento da lógica difusa aplicada à correlação de perfis)

Anonimato

Os computadores tornaram o anonimato muito mais difícil e complicado do que antes deles e da Internet. Antes, simplesmente se misturar e fazer um esforço para se parecer e agir como todo mundo já seria o suficiente para sair do radar de qualquer um que pudesse estar procurando por você — sejam "as autoridades", um cobrador, ou um vendedor insistente. Lembre-se: não é porque sabe atirar que está habilitado para uma guerra; o mais importante é sobreviver. Para um hacker é igual — não é porque sabe operar o **nmap** que está habilitado para ser hacker. O lance aqui é sobreviver, e o anonimato é a única forma para isso.

Ao longo dos anos, o anonimato na Internet tornou-se uma das questões mais cruciais, a tal ponto que hoje existe uma enorme gama de ferramentas para nos ajudar a não deixar rastros. Uma ação que venho observando é o ato de muitos alunos pularem esta parte. Esta pequena introdução, bem como o capítulo que ensina a montar ambientes seguros para o hacker, são os dois principais capítulos deste livro. Lembre-se de que, lá fora, não é só a máfia governamental que quer agir contra os hackers, mas também os próprios hackers caçam e destroem outros hackers — com autoridade posso afirmar que se chega ao ato físico e pessoal.

Por que ser invisível

A necessidade de ser invisível online não é apenas uma prerrogativa dos cibercriminosos. Em algumas partes do mundo — países governados por máfias autoritárias, e aqui se inclui o Brasil — a censura governamental é tão forte que o anonimato é necessário para não ser rastreado por serviços de espionagem públicos ou privados e para evitar penalidades. Há países onde até a pena de morte é aplicada. Na Ucrânia ou na Rússia, se você está apto para morrer em uma trincheira de guerra, será sequestrado pelo governo.

O anonimato pode ser útil para outros cenários: relatar más condições de trabalho ou políticas internas questionáveis de uma determinada empresa, ou usar a rede fora de um sistema fortemente analítico, evitando compartilhar informações sobre o que compramos ou vendemos, o que gostamos ou não gostamos, com as Grandes Empresas da Internet — escapando assim do experimento social de massa conduzido pelas grandes potências globais.

O anonimato também é uma característica fundamental para os hacktivistas, ou seja, aqueles que praticam o ativismo digital. Um exemplo é o movimento **Anonymous**, e o próprio nome reflete claramente a necessidade de não ser rastreado durante protestos online. Os hacktivistas são perigosos para o governo porque não são movidos por dinheiro — suas ações são ideológicas. Veja o autor deste livro: tem como visão hacktivista formar uma massa de agentes hackers ao difundir abertamente um conteúdo gratuito sobre o assunto, formando mais e mais hacktivistas. Dinheiro não vai parar este ser.

Se você precisa proteger sua estrutura de TI, deve considerar outro bom motivo: ser anônima como forma de prevenção, evitando qualquer exposição à Internet, onde você pode ser potencialmente afetado por qualquer pessoa. E se um especialista trabalha na área de investigação de TI, pode estar interessado em conhecer as ferramentas usadas pelos cibercriminosos para executar seus ataques permanecendo anônimos e evitando controles.

O metadado é o que importa

Quando o anonimato é pleno, a quantidade de dados obtidos sobre uma ação humana em cima do recurso tecnológico é insuficiente para a montagem de um perfil virtual — o que seria o primeiro passo para a busca de uma identidade real. No começo do capítulo foi definido que é possível criar uma identidade pelo próprio uso das tecnologias, ou seja:

- a forma como um hacker se conecta;
- os sites que visita;
- os horários de acesso;
- a forma como se expressa;
- os contatos que possui;
- e tantas outras coisas.

Mesmo que não se tenha um nome, o que importa é que o **metadado** será montado. Esse metadado será utilizado para filtrar possíveis alvos dessas grandes máfias governamentais e, dependendo do caso, até ser usado em julgamentos.

Sendo assim, o metadado de um agente que pode ser um hacker é semelhante ao mostrado abaixo. Isso porque, mesmo que o ser humano tenha conseguido omitir seu nome real, veja que o metadado é o que importa: ele bateria como uma luva no que foi descrito, e o nome poderia então ser deduzido.

```
INDIVÍDUO ea700443-e639-48f4-bf98-68de02ee85c9 = {
    "name" : "?",
    "actions_internet" : [ "escreve", "grava vídeos" ],
    "knowledge" : [ "informática" ],
    "register" : [ "google" ],
    "degree_of_anonymity" : "médio"
}
```

Mas como fugir disso? Com certeza é ter em mente que **TUDO É RASTREÁVEL**. Sabendo do problema é que podemos então coexistir com tal fato. O ato que leva ao anonimato é mais complexo do que apenas comprar um curso ou usar uma ferramenta — o autor deste livro acredita que é o ato de *ser* anônimo. Para isso, dois pilares:

- **Aleatoriedade de movimentos**
- **Pluralidade de conhecimento**

Vale entender o que há por trás dessa intuição do autor. A construção de um perfil a partir de rastros dispersos tem nome na academia e na indústria: é a **desanonimização (deanonymization)** por correlação de atributos. Ninguém precisa do seu nome — precisa de um punhado de características que, juntas, só combinam com uma pessoa. Um estudo clássico de Latanya Sweeney mostrou que a combinação de apenas três dados aparentemente inocentes — CEP, data de nascimento e sexo — identifica unicamente cerca de 87% da população dos Estados Unidos. Na Internet, os atributos são outros (estilo de escrita, horário, sites, erros de digitação), mas o princípio é idêntico: a singularidade emerge da *combinação*, não de nenhum campo isolado. É exatamente isso que os blocos JSON deste capítulo ilustram.

Aleatoriedade de movimentos

O ato mais difícil para uma máquina é a geração de um número randômico. E, por mais incrível que pareça, para um ser humano o mais difícil é ser aleatório — principalmente após anos de uso da Internet pública. Por volta de 2000–2005, um usuário comum navegava por dezenas de sites ao longo do seu dia a dia; de 2014 a 2024 (data deste livro), o usuário navega

por 2 ou 3 sites ao longo de todo o dia, o que torna mais fácil o processo de criar o metadado sobre ele.

E mesmo que uma pessoa possua mais de um perfil — ou seja, procure criar um perfil anônimo —, provavelmente terá o mesmo padrão, exatamente semelhante: visitará os mesmos sites, terá vários contatos semelhantes na rede social, cometerá os mesmos erros ortográficos, curtirá os mesmos posts ou estará nos mesmos grupos.

```
INDIVÍDUO 7a9092b6-cefd-4d9b-95d9-982784e4e954 = {
    "name" : "?",
    "actions_internet" : [ "publicar em pdf", "grava vídeos" ],
    "knowledge" : [ "informática" ],
    "register" : [ "odysee" ],
    "human_characteristics" : [ "voz grave" ],
    "degree_of_anonymity" : "médio"
}
```

Ser um hacker é ter muitos perfis, ou seja, ter muitos metadados diferentes — e quanto mais diferentes, melhor. Um hacker realiza uma campanha e, ao término dela, deve obrigatoriamente eliminar sua persona da rede mundial e abandonar suas ações. Repare que muitos grupos hackers realizam campanhas por alguns meses e somem, voltando posteriormente com outro ataque, muitas vezes com ataques semelhantes — mas são outros indivíduos. Nesses períodos de hibernação os hackers não estão dormindo como um urso; estão fazendo duas coisas:

1. Explorando para conhecer mais formas de chegar ao alvo, buscando aleatoriedade de ações;
2. Melhorando suas ferramentas ou procurando conhecer outros caminhos para chegar ao alvo.

Lembre-se de que muitos hackers morrem nesse período também — são caçados pelo que fizeram, por outros hackers ou, principalmente, pelo Estado. Acompanhe a trilogia Lazarus na playlist.

O que a escrita entrega: estilometria

Aqui é preciso ir mais fundo, porque "cometer os mesmos erros ortográficos" não é um detalhe folclórico — é uma das armas mais poderosas do lado da caça. O campo que estuda isso chama-se **estilometria (stylometry)**, a identificação de autoria pelo estilo de escrita: comprimento médio de frase, vocabulário, uso de palavras funcionais ("de", "que", "então"), pontuação, erros recorrentes, gírias, até o espaçamento depois do ponto. Cada pessoa tem uma "impressão digital linguística" tão estável que sobrevive à tentativa consciente de disfarce.

Dois casos reais ilustram o poder da técnica. O primeiro é o do **Unabomber**: por quase duas décadas, Theodore Kaczynski enviou bombas por correio nos Estados Unidos sem deixar rastro forense. O que o derrubou foi um texto — seu manifesto de 35 mil palavras, "A Sociedade Industrial e seu Futuro", publicado pela imprensa a seu pedido. Seu irmão David reconheceu o estilo, as construções de frase e certas expressões peculiares, e alertou o FBI. A análise do modo de escrever fechou o cerco; não foi a criptografia nem a rede, foi a *prosa*.

O segundo é o de **"Robert Galbraith"**: em 2013, um romance policial de estreia, "O Chamado do Cuco", assinado por esse pseudônimo, foi desmascarado como obra de **J.K. Rowling**. Após uma denúncia inicial, jornalistas do *Sunday Times* recorreram a especialistas em estilometria

(entre eles Patrick Juola e Peter Millican), que compararam por software o texto com obras de Rowling e de outras autoras. Padrões de palavras curtas e comuns, distribuição de comprimento de palavra e sequências de caracteres apontaram com força para Rowling — que confirmou. Duas lições: a máquina consegue o que o olho humano não vê, e o disfarce voluntário de estilo é frágil.

Pesquisadores do grupo de Arvind Narayanan mostraram, em 2012, que é possível identificar o autor entre mais de 100 mil autores de blogs anônimos com precisão notável apenas pelo estilo, e que técnicas de "ofuscação" de escrita ou de tradução automática ida-e-volta degradam, mas não eliminam, a assinatura. Para o hacker, a consequência é dura: *não basta esconder o nome — é preciso escrever diferente em cada persona*, e isso é muito mais difícil do que parece.

O CAPTCHA que estuda você

Vamos a um exemplo real de aleatoriedade, o famoso desafio de imagens. Responda e pense sobre a sua aleatoriedade:

- Você sempre acerta?
- Qual o tempo médio de resposta por clique?
- Você pede para ouvir?
- Pede para trocar as imagens?

Não, né? Sempre age da mesma forma. E se essa ferramenta vai além de verificar se você é um robô? E se ela identifica seus estímulos? Uma dica: sempre que criar um novo perfil, tente errar um tipo de pergunta que você normalmente não erraria — mas não utilize esse mesmo erro em outro perfil. Vamos imaginar que num perfil você nunca erra; já em outro perfil você sempre erra o barco, ou pede novas imagens quando a pergunta é "barco".

```
INDIVÍDUO e183f8aa-7eb8-4dfb-a256-590dff8bca94 = {
    "name" : "?",
    "actions_internet" : [ "escreve em pdf", "grava vídeos" ],
    "knowledge" : [ "informática" ],
    "register" : [ "odysee" ],
    "human_characteristics" : [ "voz grave" ],
    "degree_of_anonymity" : "médio",
    "captcha" : [ "rápido", "erra barcos" ]
}
```

A intuição do autor está tecnicamente correta, e vale nomear o que acontece por baixo. O **reCAPTCHA v3**, do Google, não mostra nenhum desafio de imagem: ele roda de forma invisível em todas as páginas onde está embutido e devolve ao site uma **pontuação de risco (risk score)** de 0,0 a 1,0, indicando o quão "humano" você parece. Essa nota não vem de você resolver um quebra-cabeça — vem do seu *comportamento*: a curva de movimento do mouse, o ritmo de digitação, a rolagem da página, o tempo entre ações, o histórico da sua conta Google, os cookies e a impressão digital do navegador. Em outras palavras, o CAPTCHA moderno é um coletor de **biometria comportamental (behavioral biometrics)**: a forma como você move o cursor, a cadência com que aperta as teclas e a pressão do toque na tela são tão pessoais quanto uma digital, e são usadas por bancos e plataformas para reconhecer o usuário *sem senha*. A dica do autor — variar o erro por persona — é, no vocabulário técnico, injetar ruído para impedir que essa assinatura comportamental una os seus perfis.

Pluralidade de conhecimento

As mesmas falas, os mesmos discursos... parece que estou ouvindo aquele estatista falar das mesmas coisas na minha cabeça, outra vez. A fala é a capacidade do ser humano de expor seu conhecimento. Conhecimento limitado é visível e ajuda a identificar uma pessoa; por isso, uma ação muito importante para um hacker é ter conhecimento diverso — sobretudo, saber discutir sobre tudo.

Quando montar um perfil novo, entre em fóruns temáticos por perfil, defina interesses por perfil que você tenha e interaja com a rede mundial manualmente, nos segmentos de conhecimento definidos. Para um perfil, por exemplo, vou definir que sou amante de motos: fóruns, matérias e notícias sobre o tema.

```
INDIVÍDUO 6bf466c9-a6d0-4041-acc3-b2cba933e11b = {
    "name" : "?",
    "actions_internet" : [ "escreve em pdf", "grava vídeos" ],
    "knowledge" : [ "informática" ],
    "register" : [ "odysee" ],
    "human_characteristics" : [ "voz grave" ],
    "degree_of_anonymity" : "alto",
    "captcha" : [ "rápido", "erra barcos" ],
    "interests" : [ "motos", "aventuras 2 rodas", "curtir férias de moto" ]
}
```

Evite discutir contra você mesmo com perfis diferentes, e não caia na mesma rotina de locais — há muito conhecimento para ser explorado. Repare, na evolução dos blocos JSON acima, que cada persona bem construída *acrescenta* atributos plausíveis e coerentes entre si, elevando o `degree*of*anonymity` de "médio" para "alto". O objetivo não é ter menos metadado — é impossível ter zero —, e sim ter *muitos metadados distintos e verossímeis*, de modo que nenhum deles se cruze e aponte para a mesma pessoa.

A impressão digital do navegador

Há uma armadilha silenciosa que fura toda a disciplina de personas: mesmo trocando de conta, de estilo e de assunto, se você usa a *mesma máquina* e o *mesmo navegador*, entrega uma assinatura técnica que costura tudo de volta. É o **fingerprinting** — a impressão digital do navegador. Sem cookies e sem login, um site coleta dezenas de características do seu ambiente: modelo e versão do navegador, sistema operacional, resolução de tela, fuso horário, lista de fontes instaladas, placa de vídeo (via *canvas* e WebGL), idiomas aceitos e extensões. Combinadas, essas características formam um identificador quase único.

A Electronic Frontier Foundation mantém uma demonstração pública disso, a ferramenta **Cover Your Tracks** (sucessora do Panopticlick): você a acessa e ela calcula quantos bits de entropia seu navegador expõe e se sua configuração é única entre milhares de visitantes. Na prática, quanto mais você "personaliza" o navegador — instala extensões, muda fontes, maximiza a janela num tamanho incomum —, mais raro e, portanto, mais rastreável você fica. É o mesmo paradoxo da aleatoriedade humana: o esforço para ser diferente, feito de forma consistente, vira um padrão. Por isso a defesa de rede (Tor, VPN) precisa andar junto com a defesa de ambiente (máquina virtual descartável, navegador padronizado, janela não maximizada) — tema que este livro retoma nos capítulos de ambiente seguro.

Ferramentas citadas

- **nmap** — scanner de rede e de portas, referência da área; **código aberto** (licença própria baseada na GPL).
- **reCAPTCHA v3** — serviço do Google que atribui pontuação de risco comportamental de forma invisível; **serviço comercial/proprietário** (gratuito para o site que o embute).
- **Cover Your Tracks** — ferramenta da EFF para testar a impressão digital e o rastreamento do navegador; **código aberto**.
- **Tor Browser** — navegador para roteamento anônimo, tratado em capítulo próprio; **código aberto**.

Referências

- NARAYANAN, Arvind et al. **On the Feasibility of Internet-Scale Author Identification**. IEEE Symposium on Security and Privacy (S&P), 2012.
- SWEENEY, Latanya. **Simple Demographics Often Identify People Uniquely**. Carnegie Mellon University, Data Privacy Working Paper, 2000.
- JUOLA, Patrick. **How a Computer Program Helped Show J.K. Rowling Write A Cuckoo's Calling**. Scientific American, 2013.
- FBI. **Unabomber Case** — identificação de Theodore Kaczynski pelo manifesto "Industrial Society and Its Future", 1996.
- ELECTRONIC FRONTIER FOUNDATION (EFF). **Cover Your Tracks**. Disponível em: https://coveryourtracks.eff.org/
- GOOGLE. **reCAPTCHA v3 Developer's Guide** — pontuação de risco (score). Disponível em: https://developers.google.com/recaptcha/docs/v3

Privacidade

A autenticação digital, que vimos no capítulo anterior, existe em boa parte para servir a este: proteger a privacidade. Quando eu me autentico, reduzo o risco de acesso não autorizado às minhas informações — mas quero deixar claro desde a primeira linha que privacidade **não é anonimato**. São coisas diferentes, e confundi-las é o primeiro erro de quem começa a estudar o assunto. No anonimato não existe identidade; eu opero sem que se saiba quem eu sou. Na privacidade a identidade **existe** — há um nome, um titular, uma pessoa real por trás dos dados —, mas há uma restrição sobre o acesso a esses dados. Idealmente, somente a pessoa e a solução tecnológica que ela usa possuem dados sobre a identidade daquele usuário. Ninguém mais. Este capítulo trata de manter esse círculo o menor possível, de por que ele quase sempre é rompido, e das duas formas muito diferentes pelas quais isso acontece.

Privacidade não é anonimato

Vale insistir na distinção porque ela organiza tudo o que vem depois. **Anonimato (*anonymity*)** é a ausência de vínculo entre uma ação e uma identidade: eu publico, eu acesso, eu me comunico, e não se consegue amarrar aquilo a mim. É o que os capítulos seguintes vão perseguir com o Tor, com o Whonix, com as VPNs. **Privacidade (*privacy*)** é outra coisa: aqui eu me identifico — para o banco, para a operadora, para o serviço — e o combinado é que aquele dado fique restrito àquele relacionamento. Eu digo ao banco quem eu sou; o que não quero é que o banco espalhe, venda ou entregue essa informação a quem eu não autorizei.

Repare que uma coisa não substitui a outra. Eu posso ter privacidade sem anonimato — é o caso do meu prontuário médico, que tem meu nome mas deveria estar trancado. E posso ter anonimato sem privacidade — um vazamento que expõe um perfil sem nome, mas com todos os hábitos de navegação, quebra a privacidade daquele comportamento mesmo sem saber de quem é. O hacker precisa dos dois, em momentos diferentes, e precisa saber qual está usando. Este capítulo é sobre o primeiro; os próximos, sobre o segundo.

As duas formas de violação

A violação da privacidade acontece quando dados sobre a identidade de usuários são publicados, liberados sob represália ou roubados. E ela ocorre, no fundo, de duas formas básicas — que eu separo desde já porque a defesa contra cada uma é radicalmente diferente:

- **O roubo dos dados** — alguém acessa a informação sem permissão, por uma falha de segurança.
- **A coerção estatal** — o Estado obriga quem guarda o dado a entregá-lo. Se você quer um exemplo concreto e recente, busque por *Twitter Files Brazil*.

Guarde essa divisão, porque ela é a espinha dorsal do capítulo. Contra o roubo, a defesa é **técnica**: segurança, cifragem em repouso, controle de acesso, o mínimo de dado coletado. Contra a coerção, a defesa é **arquitetural**: fazer com que o serviço simplesmente **não tenha** o que entregar — porque não coletou, ou porque coletou de forma que nem ele mesmo consegue ler. São duas engenharias distintas para dois adversários distintos, e mais adiante volto a cada uma com detalhe.

Antes disso, uma afirmação que é minha e que sustento sem meio-termo: **toda tomada de dados pessoais é ilícita.** A violação da privacidade jamais deveria ocorrer, não importa o

cenário — porque cenários podem ser distorcidos. Hoje o argumento é o terrorismo; amanhã é a pandemia; depois é a "segurança nacional". Sempre há uma narrativa boa o bastante para justificar abrir o dado de alguém, e é justamente por isso que a regra tem de ser absoluta e não circunstancial. Se a exceção depende do cenário, e o cenário é fabricável, então a exceção é a porta pela qual tudo passa.

O que são dados pessoais e PII

Antes de defender, preciso definir o que estou defendendo. Uma violação de privacidade por roubo de dados começa com uma falha de segurança e termina com a exposição ou o roubo de informação. E a informação que interessa ao atacante é o que se chama de **dado de identificação pessoal**, ou **PII (*Personally Identifiable Information*)**: qualquer dado que, sozinho ou combinado, identifique uma pessoa. Nome, endereço, número de documento — no exemplo americano clássico, o *Social Security Number* —, detalhes de cartão de crédito, e por aí vai.

Vale distinguir dois graus, porque a legislação distingue e o hacker deveria distinguir também. Há o dado **direto**, que sozinho já aponta para alguém (CPF, e-mail, telefone). E há o dado **indireto** ou **quase-identificador (*quasi-identifier*)**, que sozinho é inócuo mas, cruzado com outros, reconstrói a pessoa. O exemplo canônico é de Latanya Sweeney: **CEP, data de nascimento e sexo** bastam para identificar de forma única cerca de 87% da população dos Estados Unidos. Nenhum desses três é "sensível" isoladamente — mas juntos são uma digital. É por isso que "anonimizar" um banco de dados apagando só o nome quase nunca anonimiza nada; a re-identificação por cruzamento é uma das técnicas mais produtivas que existem.

Acima disso, as leis criam uma categoria de **dados sensíveis (*special category data*)**: origem racial, opinião política, convicção religiosa, filiação sindical, saúde, vida sexual, dados genéticos e biométricos. Esses recebem proteção reforçada exatamente porque, vazados, não expõem só a conta bancária — expõem a pessoa a perseguição. Para um hacker, o dado sensível é o que o Estado mais quer e o que ele menos pode deixar existir amarrado ao seu nome.

A defesa contra o roubo: segurança e minimização

Contra a primeira forma de violação — o roubo —, a melhor dica que eu dou a qualquer desenvolvedor é antiga e continua sendo a mais eficaz: **coletar de seus usuários o mínimo de informação possível.** O dado que você não coletou é o dado que ninguém rouba de você, que nenhum juiz obriga você a entregar, que não aparece em nenhum vazamento com o seu nome no cabeçalho da notícia. Este princípio tem nome nas leis e é o coração deste capítulo: **minimização de dados (*data minimisation*)** — coletar apenas o adequado, pertinente e limitado ao necessário para a finalidade declarada, e não guardar por mais tempo do que preciso.

A minimização se desdobra em algumas práticas concretas que valem tanto para quem constrói um serviço quanto para quem só quer entender o terreno:

- **Não peça o que não usa.** Cada campo de formulário é um passivo. Se o serviço funciona sem a data de nascimento, ela não deveria estar lá.
- **Descarte no prazo (*data retention*).** Log que você não precisa mais é munição para o adversário. Apagar é uma medida de segurança, não uma faxina.

- **Pseudonimize e separe.** Guardar o identificador em uma tabela e os dados em outra, ligadas por uma chave, faz o roubo de metade valer pouco.
- **Cifre em repouso e em trânsito.** O dado que precisa existir deve existir ilegível para quem não tem a chave.

Nada disso é teoria de manual. O vazamento da **Equifax**, em 2017, expôs dados de cerca de **147 milhões** de pessoas — nomes, números de *Social Security*, datas de nascimento, endereços e, para parte das vítimas, números de carteira de motorista e de cartão de crédito — a partir de uma única vulnerabilidade não corrigida no Apache Struts, para a qual já havia correção publicada. Uma empresa cujo negócio **é** guardar o dado financeiro alheio deixou uma porta aberta por meses. E o caso ilustra o segundo princípio da minimização: o problema não foi só a brecha, foi a **quantidade** e a **sensibilidade** do que estava acumulado atrás dela. Menos dado guardado teria significado menos dano no dia da falha. A lista de exemplos é longa e você reconhecerá os nomes — Yahoo, com bilhões de contas comprometidas entre 2013 e 2014; os cadastros nacionais que já circularam aos pedaços por aqui —, mas a lição é sempre a mesma: **o dado acumulado é um passivo, não um ativo.**

A defesa contra a coerção: não ter o que entregar

A segunda forma de violação é de outra natureza, e é aqui que a conversa fica desconfortável. O Estado, por meio do **metadado** obtido sobre as ações de um hacker, pode exercer **coerção estatal** contra os sites e serviços para obrigá-los a entregar os dados privados que ajudam a corroborar a ligação entre uma ação — dita criminosa pelo Estado — e a pessoa real. É uma forma de **ligar os fatos às pessoas**: o serviço tem o registro, o Estado tem a força para exigir o registro, e o cerco se fecha por baixo do pano, sem nunca quebrar uma única linha de criptografia.

Repare por que a defesa técnica contra roubo **não resolve** este caso. Você pode ter o melhor firewall do mundo, a melhor cifra em repouso, o controle de acesso mais rígido — e nada disso importa quando é o próprio dono da chave que recebe uma ordem judicial para abrir. Contra roubo, o inimigo é externo e você protege a fechadura. Contra coerção, o inimigo tem o mandado para exigir a chave de **você**. A fechadura não serve; o que serve é não ter o cofre, ou ter um cofre que nem você abre.

Daí a estratégia decisiva, e a diferença prática que quero que fique gravada: **um serviço só pode entregar aquilo que consegue ler.** Se ele coletou o mínimo, tem pouco a entregar. E se o que ele guarda está protegido por **criptografia forte de ponta a ponta (*end-to-end encryption*)** ou por **conhecimento zero (*zero-knowledge*)** — em que as chaves ficam com o usuário, e não com o servidor —, então a ordem judicial chega e o serviço, com toda a honestidade, responde que não tem como abrir. Não é desobediência; é impossibilidade técnica construída de propósito. Essa é a razão pela qual a criptografia forte é uma questão política, e não apenas de engenharia: ela desloca o poder de abrir o dado de quem tem a força para exigir para quem tem a chave para usar. O metadado, porém, continua sendo o ponto fraco — quem falou com quem, a que horas, de onde — e é por isso que os próximos capítulos atacam justamente o metadado de rede, que a cifra de conteúdo não protege.

Os direitos do titular: LGPD e GDPR

Existe um contrapeso jurídico a tudo isso, e um hacker precisa conhecê-lo — nem que seja para saber o que exigir de um serviço e o que denunciar quando é violado. A **LGPD (Lei Geral de Proteção de Dados, Lei nº 13.709/2018)** no Brasil e o **GDPR (*General Data Protection Regulation*)** na União Europeia partem da mesma ideia: o dado pessoal

pertence à pessoa — o **titular (*data subject*)** —, e quem o coleta e trata é apenas um guardião com deveres. Ambas consagram a minimização como princípio, exigem uma **base legal** para cada tratamento e dão ao titular um conjunto de direitos que valem a pena listar:

- **Acesso e confirmação** — saber se tratam meus dados e quais são.
- **Correção** — exigir que dados incompletos ou errados sejam corrigidos.
- **Eliminação** — pedir o apagamento; no GDPR, o célebre *direito ao esquecimento (right to erasure)*.
- **Portabilidade** — levar meus dados de um serviço para outro em formato legível.
- **Oposição e revogação do consentimento** — dizer não, e voltar atrás quando o tratamento se baseia em consentimento.
- **Informação e transparência** — saber com quem meus dados foram compartilhados.

A honestidade obriga a uma ressalva, e ela conecta as duas metades deste capítulo. A lei protege bem o titular contra o **abuso privado** — a empresa que vende o dado, que coleta demais, que vaza por negligência. Mas essas mesmas leis abrem exceções amplas para o próprio Estado, em nome de segurança pública e investigação criminal. Ou seja: o arcabouço legal ataca sobretudo a **primeira** forma de violação, o roubo e o abuso comercial, e deixa uma porta larga aberta para a **segunda**, a coerção estatal. Por isso a lei é necessária, mas não suficiente — e por isso o hacker não terceiriza a própria privacidade para o legislador. Ele conhece a lei, usa a lei, e ao mesmo tempo constrói, com criptografia e minimização, a defesa que nenhuma lei lhe dará.

Ferramentas citadas

- **Signal** — mensageria com criptografia de ponta a ponta e coleta mínima de metadados; **código aberto** (GPLv3/AGPL). Exemplo prático de "não ter o que entregar".
- **Tor** — rede de anonimato que protege o metadado de rede que a cifra de conteúdo não cobre; **código aberto** (licença BSD). Tratado em capítulo próprio.

Referências

- BRASIL. **Lei nº 13.709, de 14 de agosto de 2018 (Lei Geral de Proteção de Dados Pessoais — LGPD).** Diário Oficial da União, Brasília, 2018.
- UNIÃO EUROPEIA. **Regulamento (UE) 2016/679 do Parlamento Europeu e do Conselho (Regulamento Geral sobre a Proteção de Dados — GDPR).** Jornal Oficial da União Europeia, 2016.
- SWEENEY, Latanya. **Simple Demographics Often Identify People Uniquely.** Carnegie Mellon University, Data Privacy Working Paper 3, 2000.
- U.S. GOVERNMENT ACCOUNTABILITY OFFICE (GAO). **Data Protection: Actions Taken by Equifax and Federal Agencies in Response to the 2017 Breach.** GAO-18-559, 2018.
- NIST. **Guide to Protecting the Confidentiality of Personally Identifiable Information (PII).** Special Publication 800-122, 2010.

Tor: o Onion Router

Quando um usuário comum abre um navegador comum e acessa um site comum, o tráfego percorre uma rede complexa de roteadores, pontes e servidores intermediários — e tudo fica amarrado, de ponta a ponta. Para quem lê notícias ou usa rede social, isso é irrelevante. Para um hacker, é fatal. Este capítulo trata da primeira e mais conhecida ferramenta contra esse rastreamento: o Tor, o Onion Router. Não é uma bala de prata — e boa parte deste capítulo é justamente sobre como ele falha —, mas é uma peça que você precisa dominar.

O problema da linha amarrada

Se a comunicação usa HTTP, tudo trafega em texto plano, visível a qualquer programa espião no caminho entre o navegador e o servidor. Mesmo com HTTPS — que adiciona uma camada de criptografia entre as pontas — a informação continua **amarrada**: dá para seguir a linha da conexão até as extremidades e, com isso, saber quem lê e quem escreve. O HTTPS esconde o conteúdo da requisição, mas não esconde os **metadados** que mais interessam a quem vigia: qual endereço IP falou com qual endereço IP, a que horas, por quanto tempo e movimentando quantos bytes. Quem lembra da Pantera Cor-de-Rosa seguindo o fio entende a ideia; na prática, seguir essa "linha" virtual é ainda mais simples do que no desenho, e existe um arsenal de ferramentas para isso.

Por mais que sua conexão pareça uma entre bilhões, a quantidade atrapalha mas não impede o monitoramento de um alvo específico. Abaixo do mundo indexado e conhecido — a "internet" das redes sociais e dos sites de notícia, que só existem porque alguém permite — há um mundo que se esconde atrás de tecnologias que dificultam o rastreamento. Usando o clássico iceberg como metáfora: a **surface web** é a ponta visível; abaixo da linha d'água está a enorme massa da **deep web** (tudo o que não é indexado — o seu internet banking, um documento atrás de login, uma API privada); e dentro dela, uma fração muito menor exige um artifício tecnológico específico para ser acessada — a **darknet**. Confundir deep web com darknet é erro de iniciante: a maior parte da deep web é banal.

Como o roteamento Onion funciona por dentro

O Tor combina duas técnicas. A primeira é a criptografia da comunicação, que atrapalha os espiões entre origem e destino. A segunda é o **roteamento intermediário aleatório**: a conexão "ricocheteia" por vários pontos do planeta, e a cada salto se aplica um mascaramento de quem é a fonte real. A figura a seguir mostra um circuito de três saltos e o que um espião consegue ver em cada trecho.

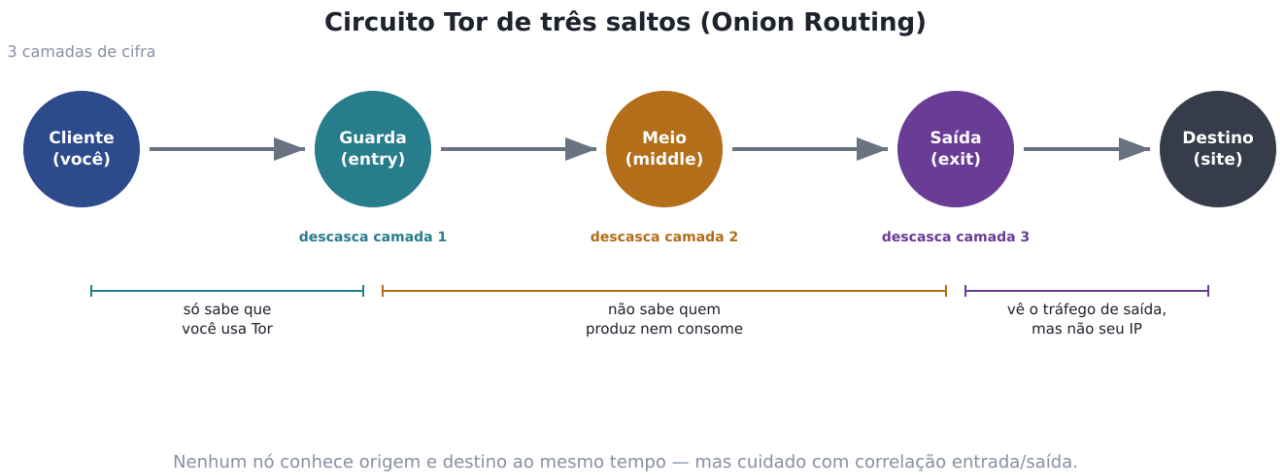

Figura: Circuito Tor de três saltos e o que cada trecho revela.

O nome "Onion" não é decorativo: descreve exatamente o que acontece com o pacote. Ao montar o circuito, o cliente negocia uma chave simétrica com **cada** um dos três relays. Depois, cada pacote é embrulhado em três camadas de criptografia — a mais interna para o nó de saída, a do meio para o nó intermediário, a mais externa para o nó de guarda. Cada relay descasca **uma** camada com sua própria chave, aprende apenas para onde encaminhar em seguida, e passa adiante. Nenhum nó sozinho conhece origem e destino ao mesmo tempo:

- **Nó de guarda (guard/entry):** conhece seu IP real, mas só vê tráfego Onion cifrado — não sabe o destino final. O Tor fixa um pequeno conjunto de guardas por meses (os *entry guards*), justamente para reduzir a chance de você, ao longo do tempo, cair num guarda malicioso.
- **Nó do meio (middle):** não conhece nem sua origem nem o destino — só repassa entre guarda e saída. É a camada que garante que guarda e saída não conversem diretamente.
- **Nó de saída (exit):** conhece o destino e vê o tráfego **como ele sai** (em texto plano, se você usou HTTP), mas não conhece seu IP real.

Traduzindo o que cada espião de rede veria, conforme onde ele está grampeando:

- **Entre o cliente e o guarda:** sabe-se apenas que você usa a rede Tor.
- **Entre relays:** sabe-se que alguém se comunica pelo Tor, mas não quem produz nem quem consome.
- **Entre a saída e o destino:** sabe-se que alguém consome aquele serviço, mas não quem.

A rede não é mágica nem anárquica: um conjunto de **autoridades de diretório (directory authorities)** — nove servidores de confiança — publica de hora em hora o **consenso**, a lista assinada de todos os relays e seus atributos. É desse consenso que seu cliente escolhe o circuito. Vale saber a origem do projeto: o roteamento Onion nasceu no **Laboratório de Pesquisa Naval dos Estados Unidos**, e o Tor recebeu por muito tempo financiamento do próprio governo americano — um paradoxo que convém ter em mente, e que alimenta a desconfiança de que parte da rede seja observada por quem a financiou.

Serviços onion e o Tor Browser

Dentro da rede publicam-se **serviços onion** (os antigos "hidden services"), com endereços `.onion` localizáveis apenas por dentro do Tor. Aqui as duas pontas ficam protegidas: nem o

cliente sabe onde está o servidor, nem o servidor sabe onde está o cliente, e o encontro se dá através de um **ponto de encontro (rendezvous point)** negociado dentro da própria rede — sem nó de saída, portanto sem o ponto fraco de saída. Os endereços evoluíram: os antigos `.onion` v2, de 16 caracteres, foram **descontinuados** por fragilidade criptográfica; os atuais v3 têm 56 caracteres (chave pública ed25519 embutida no próprio nome), e a URL enorme e sem sentido é uma vantagem — força o hacker a não inventar nomes "bonitos", que denunciam padrões mentais.

O **Tor Browser** é um Firefox ESR modificado que já inicia o Tor, direciona todo o tráfego por ele e, ao fim da sessão, apaga cookies e histórico. Ele traz três **níveis de segurança** (Padrão, Mais Seguro, O Mais Seguro) que vão desligando progressivamente JavaScript e recursos de risco — e a recomendação para quem opera é sempre subir esse nível, pagando o preço de sites quebrados. Um erro clássico que precisa ser dito: **não maximize a janela** do Tor Browser e não instale extensões; ambos criam uma impressão digital que anula o anonimato de rede.

Contra a censura: pontes e transportes plugáveis

Em países onde o próprio uso do Tor é bloqueado, a lista pública de relays é filtrada pelo provedor. A solução são as **pontes (bridges)**: relays de entrada não listados no consenso público, distribuídos sob demanda. E, quando o bloqueio é por *deep packet inspection* (que reconhece a "cara" do tráfego Tor), entram os **transportes plugáveis (pluggable transports)**, que disfarçam o tráfego: o **obfs4** o embaralha para parecer aleatório, o **meek** o esconde dentro de tráfego para grandes CDNs (domain fronting), e o **snowflake** o faz passar por uma videochamada WebRTC comum. Saber que essas camadas existem é o que separa continuar operando de ficar off-line quando o regime aperta.

Quebrando o anonimato: entrada e saída

A rede Onion é formada por "sabe-se lá quem", e não se pode confiar cegamente nela — sabe-se que Estados operam relays, principalmente os mais poderosos. A notícia boa é que entrada e saída ficam em países diferentes; a ruim é o problema dos **olhos** (as alianças de vigilância, tema de capítulo próprio). Num **ataque de confirmação de tráfego (traffic correlation)**, o adversário que observa o nó de entrada **e** o nó de saída não precisa quebrar a criptografia: basta medir o tamanho, o ritmo e a temporização dos envelopes nas duas pontas e casar os padrões. É um ataque real e documentado — a própria fundação Tor reconhece que a rede não foi projetada para resistir a um adversário global que enxergue as duas extremidades. Em 2014, pesquisadores da Carnegie Mellon (CMU/SEI) executaram um ataque de confirmação em larga escala que, segundo se noticiou, ajudou a desanonimizar usuários e serviços. Para reduzir o risco, exclui-se certos países da rota, como veremos na configuração — não elimina o ataque, mas diminui a superfície.

Quebrando o anonimato: análise de comportamento

A forma mais usada pela vigilância é relacionar o comportamento humano às ações técnicas. É fácil monitorar alguém fisicamente e, em paralelo, monitorar sua atividade virtual — e então cruzar as duas. Mas mesmo sem o corpo físico, dá para montar um metadado só pela conexão, observando a taxa de cliques por hora, o tamanho da carga de dados e a sequência dessas cargas. Junte a isso a **estilometria** (o estilo de escrita) e os padrões de horário — o fuso de quando você está "online" entrega a região do planeta — e o cerco se fecha sem nunca tocar na criptografia. No capítulo de programação será mostrada uma ferramenta para

embaralhar o uso do Tor, gerando requisições falsas aleatórias e até respondendo automaticamente em chat, elevando o ruído sobre o seu sinal. Vale ainda lembrar que uma simples conexão HTTP entrega versão de navegador, resolução de tela e sistema operacional — por isso a recomendação de trabalhar sempre em máquina virtual e variar essas características.

Quebrando o anonimato: vazamento de DNS

Antes de acessar um site, seu computador precisa traduzir o nome (por exemplo, `google.com`) em um endereço IP, e faz isso por um protocolo chamado **DNS**. Se você está seguro para o tráfego HTTP/HTTPS, precisa estar seguro também para essa consulta. Em teoria, o Tor Browser cuida disso passando tudo para o serviço Tor local; o problema aparece quando você usa **outros** aplicativos, ou o Tor no nível do sistema, e alguma biblioteca resolve o nome pelo resolvedor do provedor "por fora" do túnel. O resultado é um **vazamento de DNS**: mesmo com todo o resto no Tor, a lista do que você quis acessar fica registrada nos logs de um servidor DNS — trilha que policiais, hackers ou qualquer um com acesso privilegiado à rede pode seguir.

O caso do navegador **Brave** ilustra bem: entre 2020 e 2021, uma atualização do bloqueador de anúncios expôs as consultas DNS dos usuários do modo Tor do navegador, sem que a empresa percebesse — não só domínios `.onion`, mas toda requisição das abas Tor. A validação simples é usar o comando `nslookup` e confirmar que quem responde é um resolvedor da rede protegida (num ambiente Whonix, um endereço da faixa `10.152.152.x`), e não o resolvedor do seu provedor:

```
nslookup exemplo.com
# Server: 10.152.152.10  → OK, resolvedor interno do Whonix
# Server: 192.168.0.1    → PERIGO, você está vazando DNS para o roteador/ISP
```

A conclusão de fundo é a que o próprio manuscrito defende e que este livro reforça no capítulo do Whonix: **DNS criptografado ou tunelado não deveria ser responsabilidade do navegador, e sim do sistema operacional** — porque assim vale para todos os processos, não só para as abas do browser.

Quebrando o anonimato: injeção de malware

Se o computador do hacker não estiver bem isolado, o ataque pode se desenrolar de três formas: injeção de um **trojan** para controle remoto, injeção de um **worm** para movimentação lateral, ou execução de um artefato de código (JavaScript ou até no lado do servidor). O risco vale tanto para quem acessa quanto — e principalmente — para quem hospeda serviços `.onion`.

O caso **Freedom Hosting** é o exemplo canônico. Era o maior serviço de hospedagem `.onion` em 2013 quando o FBI, após prender o operador, injetou nas páginas hospedadas um exploit conhecido como *NIT (Network Investigative Technique)*: ele explorava uma vulnerabilidade de JavaScript em versões antigas do Tor Browser — o **NoScript** não vinha habilitado por padrão — para fazer o navegador da vítima "ligar para casa" e revelar o endereço MAC, o IP real e o nome do computador Windows aos investigadores. O proprietário, Eric Eoin Marques, foi preso na Irlanda em agosto de 2013. A lição técnica é dura e vale gravar: **uma única brecha no navegador anula toda a proteção de rede.** É por isso que se separa o navegador da pilha de rede (Whonix) e se sobe o nível de segurança do Tor Browser até desligar o JavaScript.

Não confie no nó de saída

Há um mito perigoso de que "no Tor tudo é criptografado". Não é. O nó de saída vê o seu tráfego **exatamente como ele sai** — e se você usou HTTP, ele lê tudo. Em 2007, o pesquisador Dan Egerstad montou alguns nós de saída e coletou senhas e e-mails de dezenas de embaixadas e órgãos governamentais que trafegavam credenciais em texto plano por cima do Tor. A moral: o Tor anonimiza a **origem**, não protege o **conteúdo** de saída. Use sempre HTTPS ponta a ponta (ou serviços `.onion`, que não têm saída), e nunca trafegue credenciais em texto plano — presuma que o nó de saída é hostil, porque frequentemente é.

Opsec: por que ferramenta não salva ninguém

O Tor bem usado é forte; o operador desatento é o elo que quebra. O caso de **Ross Ulbricht**, o "Dread Pirate Roberts" do Silk Road, é o estudo de caso obrigatório: o mercado rodava como serviço `.onion`, mas ele foi ligado a ele por um rastro de erros de **opsec** anteriores — um post antigo num fórum divulgando o site associado ao seu e-mail pessoal, uma pergunta no Stack Overflow feita com o nome real e depois trocada, encomendas de documentos falsos. E foi preso numa biblioteca **com o notebook ligado e destravado**, para que o FBI capturasse a sessão aberta e as chaves em memória. Nenhuma dessas falhas é da rede Tor; todas são humanas. Guarde a regra: a criptografia protege o que você faz certo, não o que você faz de bobagem.

Instalando e configurando um Tor básico

Apesar de toda a controvérsia, o Tor é uma ferramenta que você precisa saber operar — apenas uma entre várias que usará para elevar seu grau de anonimato. No Debian a instalação é direta, mas a configuração exige atenção. Comece atualizando e instalando:

```
sudo apt update -y
sudo apt install tor -y
```

O Tor vem com um arquivo de configuração padrão. Edite `/etc/tor/torrc` e acrescente, ao final, um bloco como o abaixo — lembrando de manter a lista de países excluídos sempre atualizada, porque a geopolítica muda:

```
RunAsDaemon 1
TransPort 9040
SocksPort 9050
HTTPTunnelPort 9060

ExcludeNodes {us},{gb},{ca},{au},{nz},{fr},{de},{nl},{no},{dk},{be},{se},{es},{it}
ExcludeExitNodes {us},{gb},{ca},{au},{nz},{fr},{de},{nl},{no},{dk},{be},{se},{es},{it}
StrictNodes 1

VirtualAddrNetwork 10.192.0.0/10
AutomapHostsOnResolve 1
DNSPort 53
```

Linha a linha: `RunAsDaemon 1` mantém o Tor como serviço na inicialização. O `TransPort` abre a porta 9040 para tráfego de transporte comum, e o `SocksPort` a clássica 9050 para **SOCKS5**; como muitas tecnologias não falam SOCKS5, o `HTTPTunnelPort` habilita também um proxy HTTP na 9060. As diretivas `ExcludeNodes` e `ExcludeExitNodes` evitam os países das alianças de vigilância — sendo a saída (`ExcludeExitNodes`) a mais crítica, porque é lá que o tráfego se revela —, e o `StrictNodes 1` torna essa exclusão **obrigatória** (sem ele, o

Tor pode ignorar a lista se não achar rota). Por fim, `VirtualAddrNetwork`, `AutomapHostsOnResolve` e `DNSPort` preparam o terreno para resolver DNS **dentro** do Tor, mapeando nomes para endereços virtuais e escutando consultas na porta 53 — a base para eliminar o vazamento de DNS visto acima.

Depois de salvar, reinicie e confirme o serviço:

```
sudo systemctl restart tor
sudo systemctl status tor
```

Uma advertência honesta, que vale mais que qualquer configuração: excluir muitos países deixa a navegação lenta e pode fazer alguns endereços `.onion` retornarem erro. E, por mais tentador que seja, **não baixe torrents pelo Tor** — o cliente BitTorrent costuma anunciar seu IP real por fora do proxy e ainda esmaga a rede, que é um bem comum e escasso. Para torrent, VPN ou I2P, como veremos nos próximos capítulos.

Ferramentas citadas

- **Tor** — rede de roteamento anônimo (roteamento Onion); **código aberto** (licença BSD de 3 cláusulas).
- **Tor Browser** — Firefox ESR modificado com Tor, NoScript e ajustes de privacidade; **código aberto**.
- **NoScript** — extensão que bloqueia scripts não confiáveis por origem; **código aberto** (GPL).
- **obfs4 / meek / snowflake** — transportes plugáveis para driblar censura ao Tor; **código aberto**.

Referências

- DINGLEDINE, Roger; MATHEWSON, Nick; SYVERSON, Paul. **Tor: The Second-Generation Onion Router**. USENIX Security Symposium, 2004.
- THE GUARDIAN. **Attacking Tor: how the NSA targets users' online anonymity**. 2013.
- O'BRIEN, Sean (ExpressVPN Digital Security Lab). Análise do vazamento de DNS do navegador Brave, 2021.
- BIRYUKOV, Alex; PUSTOGAROV, Ivan; WEINMANN, Ralf-Philipp. **Trawling for Tor Hidden Services**. IEEE S&P, 2013.
- TOR PROJECT. **Tor: Pluggable Transports** e **Onion Services v3**. Disponível em: https://community.torproject.org/

Whonix

Quando um usuário consome dados pela rede Tor este pode ser localizado. A rede Tor não dá um anonimato pleno e nem ajuda o usuário a chegar o mais próximo possível da perfeição neste quesito: um JavaScript mal-intencionado desenvolvido para localizar dados pode expor o usuário, por isso as regras sobre JavaScript quando se utiliza o Tor Browser são tão pesadas. Um programa mal-intencionado pode ser executado — quem nunca clicou em um link ou em uma imagem? O capítulo anterior mostrou que quase toda a superfície de ataque sobrevivente ao Tor mora **fora** da rede: no navegador, nas outras aplicações, no sistema operacional que resolve DNS por fora do túnel. A pergunta natural, então, é: e se a máquina inteira fosse construída de tal modo que nada — nenhum processo, nenhuma biblioteca, nenhum malware — conseguisse falar com a internet a não ser pelo Tor? Essa é a proposta do Whonix.

O Whonix-Gateway força a única saída

O **Whonix-Gateway** é um software projetado para executar o roteamento Onion visto até aqui; ele força toda a conexão vinda de uma outra máquina virtual a passar obrigatoriamente pelo Tor. Ou seja, toda a comunicação — nenhuma escapa deste roteamento. O Whonix não muda como a rede Tor funciona, mas garante que ela seja a **única saída** para as máquinas isoladas atrás do Whonix-Gateway. Mais uma maravilha do mundo cypherpunk.

A diferença de fundo em relação a instalar o Tor "por dentro" de uma máquina, como fizemos no capítulo anterior, é conceitual e vale gravar: no Tor local, quem garante que o tráfego vai pelo Tor é a **configuração** — um proxy, um `torrc`, uma disciplina de rota. Se uma aplicação ignorar o proxy, se uma biblioteca resolver DNS por fora, se um exploit abrir um socket direto, o tráfego vaza e o operador sequer percebe. No Whonix, essa garantia não vem da configuração da aplicação: vem da **topologia da rede**. A máquina onde você trabalha simplesmente não tem outra rota física para o mundo. Ela só enxerga um gateway, e esse gateway só sabe falar Tor. É a diferença entre pedir para o tráfego se comportar e tornar o mau comportamento fisicamente impossível.

Por baixo, o Whonix é construído sobre o **Kicksecure**, uma distribuição de segurança endurecida derivada do **Debian GNU/Linux**. Não é um sistema exótico feito do zero: é Debian com um conjunto grande de mitigações de segurança aplicadas de fábrica (kernel endurecido, permissões restritas, remoção de superfície de ataque) e, sobre isso, a camada de força-Tor que caracteriza o Whonix. Herdar o Debian é uma vantagem prática — você tem o `apt`, os mesmos pacotes, a mesma documentação enorme — com o hardening já resolvido por quem projetou o sistema.

Duas máquinas virtuais: Gateway e Workstation

Sempre que se faz o download do Whonix, é realizado o download de duas **Virtual Machines (VM)**. É um ambiente que não deve ser instalado no metal: uma VM é de fácil exclusão e nunca é fixa — é mais fácil apagar e reconstruir uma VM do que fazer uma instalação física do zero. Nesse download vêm duas máquinas virtuais. Uma é a **Whonix-Gateway**, que é a máquina que faz toda a mágica acontecer. A outra é a **Whonix-Workstation**, que é uma máquina com algumas particularidades para o usuário utilizar — e esta segunda pode ser descartada e substituída por qualquer outra distribuição, inclusive o **Kali GNU/Linux**.

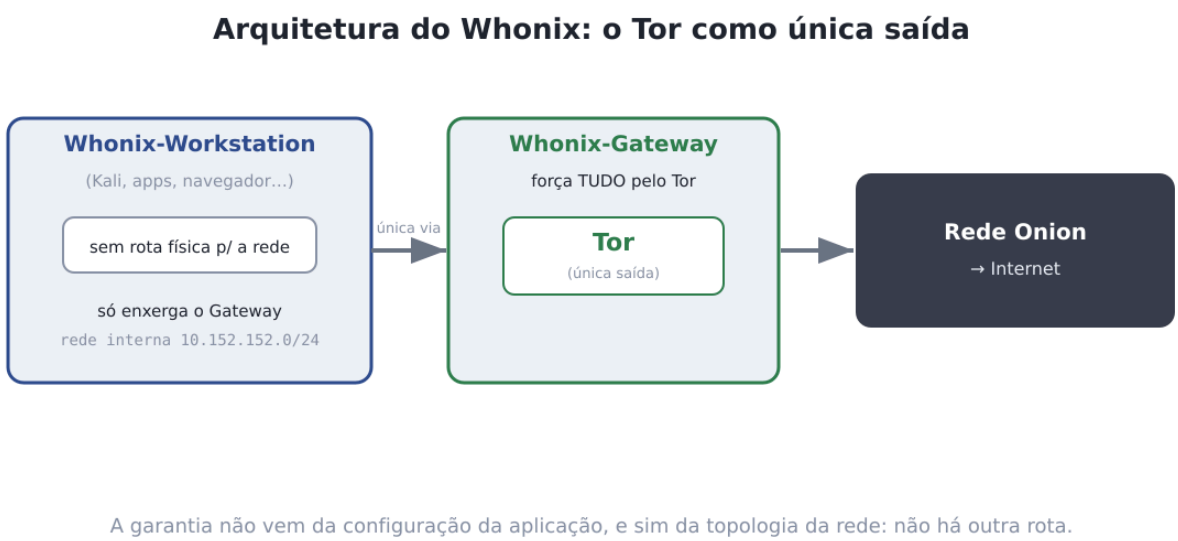

Figura: Arquitetura do Whonix — o Tor como única saída.

A geometria dessa separação é o coração do projeto, então vale detalhá-la. As duas VMs são ligadas por uma rede **interna e isolada**: a Workstation possui apenas uma interface de rede, e essa interface conversa exclusivamente com o Gateway — normalmente numa faixa privada como `10.152.152.0/24`. A Workstation **não tem** um caminho direto para a placa de rede física do computador. Quem tem contato com o mundo real é só o Gateway, através de uma segunda interface. Assim, quando um programa na Workstation quer acessar `algumsite.com`, o pacote não tem para onde ir a não ser para o Gateway, e o Gateway só sabe fazer uma coisa com ele: empurrá-lo para dentro do Tor. Se observar o esquema apresentado no capítulo do Tor, percebe que aqui todos os programas, todas as conexões, todas as portas — ou seja, tudo — obrigatoriamente passa pelo Whonix-Gateway, garantindo 100% de certeza de que tudo que você faz estará na rede Tor.

Essa arquitetura resolve de raiz o problema do **vazamento de DNS** que vimos no capítulo anterior. A Workstation nem sequer conhece um resolvedor DNS "de verdade": ela pergunta ao Gateway, e o Gateway resolve os nomes por dentro do Tor. Foi por isso que, ao mostrar o comando `nslookup`, o endereço "seguro" esperado era da faixa `10.152.152.x` — é literalmente o resolvedor interno do Whonix respondendo pelo Gateway, e nunca o resolvedor do seu provedor. Aqui se cumpre, concretamente, a tese que o próprio manuscrito defende: DNS tunelado é responsabilidade do **sistema/da rede**, não do navegador.

O problema que sempre existiu: a máquina ainda pode cair

Mas mesmo assim ainda há um problema — um problema que sempre existiu em todos os cenários, inclusive neste. Um hacker com conhecimento pode invadir qualquer máquina, inclusive a Kali GNU/Linux no cenário acima. É importante ser honesto sobre o que o Whonix garante e o que ele não garante: ele garante que a máquina invadida **não conseguirá revelar o seu IP real pela rede**, porque não existe rota fora do Tor por onde vazar. Ele **não** impede, por si só, que a máquina seja comprometida, que dados dentro dela sejam roubados, ou que o atacante persista ali. O isolamento protege o anonimato de rede; não substitui o endurecimento da própria Workstation. Por isso, recomendações podem ser feitas para dificultar:

- **Trocar o MAC Address** constantemente das VMs;
- **Apagar e recriar** o cenário constantemente;

- **Senhas fortes** e naturalmente trocadas constantemente nas VMs — contrariando o **NIST** (deixe o NIST para serviços web);
- **Trabalhar as IPTABLES/NFTABLES** da máquina Kali GNU/Linux, sendo o **ufw** uma boa opção.

Vale um comentário sobre a terceira recomendação, porque ela contraria de propósito uma orientação famosa. As diretrizes do NIST (a publicação **SP 800-63B**) recomendam, para serviços web de uso massivo, **parar de forçar troca periódica** de senha e parar de exigir a complexidade artificial (aquele "mínimo um número, uma maiúscula e um símbolo"), porque na prática isso empurra o usuário comum para senhas piores e previsíveis. Essa lógica faz sentido para o site de um banco com milhões de clientes leigos. Não faz sentido aqui. No cenário do hacker, o operador é o próprio administrador, não há atrito de suporte, o adversário é dirigido e não estatístico, e a máquina é justamente o que se quer proteger. Por isso a orientação se inverte: senha longa, forte e trocada com frequência. O NIST otimiza para o comportamento do usuário médio; você não é o usuário médio.

Stream isolation: por que cada aplicação merece o seu próprio circuito

Se você instalar aplicativos personalizados e não tomar precauções explícitas contra a **correlação de identidade (identity correlation)** por meio do compartilhamento do circuito do Tor, você corre o risco de que atividades diferentes — digamos, o Google Chrome e o **IRC** — passem pelo mesmo circuito Tor e saiam no mesmo relé. Mesmo que você ainda seja anônimo, ou seja, mesmo que o relé de saída único não saiba o seu IP nem a sua localização real, o adversário pode facilmente correlacionar essas atividades emitidas por aplicativos diferentes ao mesmo pseudônimo e mapear um indivíduo com particularidades. Veja: com essas informações eu sei que você usa Google Chrome em `1280x720` e também usa IRC; eu sei que você existe, mas não sei onde você está.

Para entender por que isso é perigoso, é preciso enxergar o mecanismo. Um circuito Tor é uma linha compartilhada: enquanto ele estiver de pé, todo o tráfego que sai por ele emerge pelo **mesmo nó de saída**, no mesmo IP de saída, ao mesmo tempo. Se o seu navegador e o seu cliente de chat compartilham o circuito, quem observa aquele nó de saída vê duas coisas saindo juntas do mesmíssimo ponto: o login anônimo do chat e a navegação. Ele não sabe quem você é, mas sabe que "o usuário X do chat" e "quem abriu aquelas páginas" são **a mesma pessoa** — porque saíram amarrados. Some a isso os detalhes que uma conexão entrega de graça (a resolução de tela, a versão do navegador, o *user-agent*) e o adversário monta um perfil singular. Anonimato sem *unlinkability* — sem a impossibilidade de ligar as suas atividades umas às outras — é meio anonimato.

A defesa se chama **stream isolation** (isolamento de fluxos), e a ideia é elegante: em vez de deixar tudo compartilhar um circuito, força-se cada aplicação (ou cada finalidade) a receber o **seu próprio circuito Tor**, com o seu próprio nó de saída. O mecanismo concreto é o cliente Tor oferecer **portas SOCKS distintas**: cada `SocksPort` diferente corresponde a um circuito diferente. O navegador fala com uma porta, o cliente de IRC fala com outra, o gerenciador de pacotes fala com uma terceira — e cada um sai por um relé diferente, num IP diferente. Como resultado, o observador do nó de saída do IRC não vê a sua navegação (ela está saindo por outro relé, em outro lugar do planeta), e vice-versa. As duas atividades deixam de ser correlacionáveis pelo circuito, porque simplesmente não compartilham mais nenhum ponto.

O Whonix implementou proteção contra correlação de identidade por meio do compartilhamento de circuitos Tor. O Whonix configura a maioria dos aplicativos que vêm pré-

instalados com ele para usar `SocksPort` diferentes — portanto, nenhuma correlação de identidade fica em risco. Este é, de novo, o tema central deste capítulo: a proteção não depende de o usuário lembrar de fazer a coisa certa; ela vem **configurada de fábrica na arquitetura**. O operador não precisa saber o que é stream isolation para estar protegido por ele — e é exatamente isso que torna a defesa confiável. Uma proteção que depende da disciplina humana a cada sessão vai falhar em algum dia cansado; uma proteção embutida na topologia da máquina não tem "dia cansado".

Esse recurso não é gratuito, e é justo dizer o preço. Como cada aplicação sai por um circuito próprio, você aparece na internet comum como muitos IPs de saída diferentes ao mesmo tempo, o que aumenta a incidência de **CAPTCHA** e reduz o tempo de resposta de uma navegação — ela fica mais lenta. Mas é a solução ideal: o custo é conveniência, e o que se compra com ele é a impossibilidade de amarrar as suas identidades umas às outras.

Whonix ou Tails: isolamento persistente contra amnésia descartável

Vale posicionar o Whonix ao lado da outra ferramenta de anonimato de sistema mais conhecida, o **Tails (The Amnesic Incognito Live System)**, porque as duas resolvem problemas parecidos com filosofias opostas, e escolher errado é comum. O Tails é um sistema **live** e **amnésico**: você o carrega de um pen drive, ele roda inteiro na memória RAM, força todo o tráfego pelo Tor e, ao desligar, **esquece tudo** — não deixa rastro no computador usado, porque nada é gravado em disco. Sua força é não deixar vestígio físico e ser portátil: você senta em qualquer máquina, opera e some. O Whonix, ao contrário, é **persistente** e mora em máquinas virtuais no seu próprio computador; sua força não é o esquecimento, e sim o **isolamento** rigoroso entre a Workstation (onde você trabalha) e o Gateway (que fala com o mundo).

A regra prática para escolher: use o **Tails** quando o valor está em não deixar rastro no hardware e em poder operar de qualquer lugar sem instalar nada — o cenário do jornalista ou da fonte que precisa de amnésia e portabilidade, e para quem persistir arquivos é um risco, não um recurso. Use o **Whonix** quando você precisa de um ambiente de trabalho persistente, com ferramentas instaladas, isolamento forte e a garantia de que um comprometimento da máquina de trabalho não expõe o IP real — o cenário do operador que passa horas numa Workstation Kali atrás do Gateway. As duas abordagens, aliás, se combinam: existe quem rode o Whonix **dentro** do Tails, casando a amnésia do live com o isolamento das duas VMs. O ponto que interessa a este capítulo é que ambas fazem a mesma aposta de fundo — tirar o anonimato das mãos do usuário e depositá-lo na arquitetura do sistema.

Por que a arquitetura vence a disciplina

Se há uma lição que atravessa este capítulo e o anterior, é esta. No capítulo do Tor, quase todas as quebras de anonimato que estudamos — o vazamento de DNS por uma aplicação distraída, a janela do navegador maximizada, o download de torrent anunciando o IP real, o exploit do Freedom Hosting acionando o navegador para "ligar para casa", a prisão de Ross Ulbricht com o notebook destravado — **não foram falhas da criptografia**. Foram falhas humanas ou de configuração. A rede Tor fez o seu papel; o elo que arrebentou foi sempre o operador ou o ambiente ao redor dele.

O Whonix é a resposta de engenharia a esse padrão. Em vez de pedir ao operador que se lembre, a cada ação, de não vazar DNS, de isolar fluxos, de nunca abrir um socket direto, de conferir se aquela aplicação nova respeita o proxy — ele **remove a possibilidade de errar**. Não existe rota fora do Tor porque a Workstation não tem placa de rede física. Não existe

vazamento de DNS porque não existe resolvedor local. Não existe correlação por circuito compartilhado porque o stream isolation já vem ligado. Um exploit que caia na Workstation e tente abrir uma conexão direta encontra apenas o Gateway, que o joga no Tor de novo. A defesa não é uma regra que você segue; é uma parede que existe quer você preste atenção ou não. A disciplina humana falha porque é humana. Uma parede não tem dia ruim — e é por isso que o isolamento na arquitetura supera, e deve substituir, a boa vontade do usuário.

Ferramentas citadas

- **Whonix** — sistema de anonimato baseado em Kicksecure/Debian, composto de duas VMs (Gateway e Workstation) que forçam todo o tráfego pelo Tor e implementam stream isolation; **código aberto** (majoritariamente GPLv3).
- **Kicksecure** — distribuição Debian endurecida que serve de base ao Whonix; **código aberto** (GPLv3).
- **Kali GNU/Linux** — distribuição voltada a testes de intrusão, usável como Workstation atrás do Gateway; **código aberto** (GPL e licenças diversas).
- **Tails** — sistema live amnésico que força o tráfego pelo Tor e não persiste dados; **código aberto** (GPLv3).
- **ufw (Uncomplicated Firewall)** — front-end simplificado para iptables/nftables, para endurecer a Workstation; **código aberto** (GPLv3).
- **Tor** — rede de roteamento Onion que o Whonix usa como única saída; **código aberto** (licença BSD de 3 cláusulas).

Referências

- WHONIX. **Whonix Documentation: Design and Goals**. Disponível em: https://www.whonix.org/wiki/Documentation
- WHONIX. **Stream Isolation**. Disponível em: https://www.whonix.org/wiki/Stream_Isolation
- WHONIX. **Whonix versus Tails**. Disponível em: https://www.whonix.org/wiki/Comparison*with*Others
- KICKSECURE. **Kicksecure — A Security-hardened Linux Distribution**. Disponível em: https://www.kicksecure.com/
- TAILS. **Tails — The Amnesic Incognito Live System**. Disponível em: https://tails.net/
- GRASSI, Paul A. et al. **NIST Special Publication 800-63B: Digital Identity Guidelines — Authentication and Lifecycle Management**. National Institute of Standards and Technology, 2017.
- DINGLEDINE, Roger; MATHEWSON, Nick; SYVERSON, Paul. **Tor: The Second-Generation Onion Router**. USENIX Security Symposium, 2004.

VPN Privada

Depois de dominar o Tor, chega a hora de olhar para a ferramenta que quase todo mundo confunde com ele: a **VPN (Virtual Private Network)**. Uma boa alternativa é o uso de uma VPN privada, pois a rede Tor tem problemas já discutidos ao longo deste material. Antes de qualquer coisa, porém, é preciso deixar clara uma distinção que salva ou condena um hacker: uma VPN **não** entrega o mesmo anonimato que o Tor. São ferramentas com propósitos diferentes, e trocar uma pela outra por preguiça é o tipo de erro que enche presídio. Este capítulo mostra como usar uma VPN a favor da sua privacidade, onde ela falha, e como construir uma rede de segurança para o momento em que ela falhar — porque ela vai falhar.

Por que recorrer à VPN se já temos o Tor

A rede Tor carrega fragilidades que já levantei antes e que vale reunir aqui, porque são elas que empurram o hacker para uma solução alternativa:

- Pode haver nós espiões infiltrados nas bordas da rede Tor. Lembre-se de que é uma rede de origem governamental.
- Um nó de saída pode estar sendo utilizado por um crime de terrorismo, e esse mesmo nó é usado por uma requisição sua.
- Excesso de **Captcha**, pois há um alto *throughput* (vazão) de saída nos nós da rede Tor, e os grandes sites tratam esses IPs como suspeitos por padrão.

No caso de uma VPN privada, a empresa é responsável pelos nós. Logo, ela possui nós privados para seus clientes. É natural que muitos clientes que atuam no terrorismo possam contratar esses serviços, mas esse público atua mais na rede Tor, o que deixa os IPs da VPN comercialmente "mais limpos" — menos marcados, menos bloqueados, menos afogados em Captcha.

O que a VPN faz e o que ela não faz

Aqui preciso ser honesto com você, porque a indústria de VPN gasta fortunas de marketing dizendo o contrário. Uma VPN cria um **túnel (tunnel)** criptografado entre a sua máquina e um servidor da empresa. A partir daí, todo o seu tráfego sai para a Internet com o **IP** do servidor, e não com o seu. Isso resolve dois problemas reais: esconde o seu conteúdo do seu provedor de acesso (o **ISP — Internet Service Provider**) e troca o seu endereço aparente. Só isso. E "só isso" é muito menos do que o Tor oferece.

A diferença estrutural é de confiança. No Tor, o roteamento se espalha por três relays operados por gente diferente, e **nenhum** deles conhece origem e destino ao mesmo tempo. Numa VPN, existe **um único** ponto: a empresa. Esse ponto vê tudo — o seu IP real de um lado e o destino do outro, simultaneamente. Você não distribui a confiança, você a concentra. Se o Tor distribui o risco entre três desconhecidos, a VPN aposta tudo em um só, e você tem que rezar para que esse um seja honesto, competente e esteja numa jurisdição que não o obrigue a te entregar.

Três coisas decorrem disso, e é bom gravá-las:

- **O provedor vê tudo.** Do lado dele, o túnel termina. Se ele quiser (ou for obrigado a) registrar para onde você foi, ele pode. A promessa de "zero log" é uma promessa contratual e política, não uma garantia matemática como a do roteamento Onion.

- **Jurisdição e logs andam juntos.** De nada adianta um "sem log" no papel se a empresa está sediada num país que pode, por ordem judicial, forçá-la a começar a registrar amanhã. Por isso a escolha do país da empresa importa tanto quanto a tecnologia.
- **É um ponto único de correlação.** Um adversário que observe a entrada e a saída daquele servidor faz a ligação sem quebrar criptografia nenhuma — o mesmo problema de correlação de tráfego que aflige o Tor, só que agora concentrado num alvo único e conhecido.

A regra prática que uso: **VPN é para privacidade e para trocar de IP; Tor é para anonimato.** Quando o jogo é sério, os dois se combinam (Tor por cima da VPN, por exemplo), mas nunca troco um pelo outro fingindo que são a mesma coisa.

A imagem abaixo exibe a diferença de um **PING** sem VPN e um PING com VPN. É um preço que se paga pela sua segurança: cada pacote agora dá uma volta a mais e é cifrado no caminho, então a latência sobe.

Neste material não só utilizaremos o Tor, como também uma VPN privada, demonstrando o uso de ambas para as mais diversas funções de um hacker. Três empresas despontam, e todas oferecem privacidade para seus clientes:

- **ExpressVPN**;
- **NordVPN**;
- **CyberGhost** — será a escolhida, pois o preço está acessível.

ATENÇÃO: Não estou ganhando nenhum centavo pela indicação.

ATENÇÃO: Verifique se há disponibilidade para o **seu** sistema operacional.

Após a contratação do serviço da CyberGhost, você entrará em uma interface de painel. Recomendo contratar pacotes de 1 ou 2 anos, pois sempre sai mais barato. O primeiro passo é baixar um pacote de instalação para o sistema operacional que pretende utilizar. Esse instalador é por terminal, então pode ser usado em um GNU/Linux gráfico ou em modo texto. Para este material será utilizada uma versão para Kali GNU/Linux. Após o download, o arquivo será enviado para o Kali GNU/Linux, que será instalado no próximo capítulo.

Comparação entre VPNs

Quando se compara uma solução de VPN, deve-se analisar muitas características, mas cito as principais:

- Sem log de acessos a sites na Internet;
- Número de servidores, para garantir a renovação de IPs em casos específicos;
- Número de países;
- Ter servidores em países fora dos tratados dos **5 olhos**, **9 olhos** e **14 olhos**.

Imagine que seja possível saber o que um cliente acessa (os sites web). É natural que isso vá contra todos os princípios pregados aqui. A privacidade seria quebrada, e isso não é aceitável para hackers. Geralmente localizamos serviços com as seguintes características positivas sobre log de acesso:

- **Zero logs**;
- Sem log de acessos web.

Aqui cabe um alerta que o marketing nunca faz: "zero log" é uma **afirmação**, não uma prova. O cliente não tem como inspecionar os servidores do provedor. O que aumenta a credibilidade

da alegação são fatos externos — auditorias independentes publicadas, e sobretudo casos reais em que uma autoridade pediu dados de um usuário e a empresa **não tinha nada para entregar**. Sem isso, o "zero log" vale o que vale qualquer texto de anúncio.

Geralmente precisamos trocar o número do IP, principalmente se o hacker atua com extração de dados na web (**scraping**). Uma grande quantidade de servidores proporciona o ambiente ideal para isso, mas também deve-se observar em quais países esses servidores estão alocados. Existem organizações e tratados internacionais para troca de dados sobre cidadãos que lutam contra as máfias (governos). Esses tratados serão discutidos em um capítulo conspiracionista à parte. Trata-se do conhecido **5 olhos (Five Eyes)**: o tratado fechado por 5 países iniciais troca dados em minutos, sem necessidade jurídica. Depois o tratado foi expandido para 9 e depois para 14, mas esses últimos dentro de restrições, incluindo a necessidade de autorização judicial. A lógica para o hacker é simples: se o servidor de saída da sua VPN está dentro dos olhos, o esforço para cruzar os dados dele com os do seu provedor de origem cai drasticamente.

Router com DD-WRT: tirando o fator humano

Quanto ao uso de VPN ou Tor, um dia você vai errar. Isso é certo, e você vai naturalmente se expor. Uma ação que tive que tomar foi rodar um Linux no **router** (roteador) diretamente, e nesse Linux já deixar uma configuração de VPN pronta. Dessa forma, a chance de esquecer de ativar tal recurso deixa de depender do ser humano, que é falho.

Esse é o ponto filosófico do capítulo inteiro, e vale antes de qualquer comando. A maioria dos vazamentos não vem de criptografia quebrada; vem de gente que esqueceu de ligar a proteção, ou que reiniciou a máquina e a VPN não subiu junto, ou que abriu o notebook no café e clicou no navegador antes de conectar. Quando a VPN vive **dentro do roteador**, o túnel sobe com o aparelho, antes de qualquer dispositivo da casa tocar na Internet. Você não tem um botão para esquecer de apertar. Todo dispositivo que passar por aquele roteador — notebook, celular, o que for — sai pela VPN por padrão, sem software cliente em cada um. Tirar o fator humano do circuito é, quase sempre, a melhor medida de segurança que existe.

DD-WRT é um Linux com poucos megabytes; um router com 4 MB já suporta essa distribuição. Porém, para ter a VPN ativa é preciso ter 8 MB de memória flash, então, para o nosso uso, precisamos encontrar determinados modelos. Para este exemplo vou utilizar o **Linksys EA6500**, um aparelho que possui recursos suficientes para rodar o DD-WRT com **OpenVPN Client**. (Encontrei routers com 8 MB que não possuem o OpenVPN — isso carece de confirmação.)

O trabalho mais difícil é localizar dispositivos compatíveis, pois praticamente todos os itens vendidos no Brasil são destinados ao submundo do consumo de massa e não são compatíveis com o DD-WRT. É sério: esse Linux é feito e configurado para o **chipset**, então temos restrições pesadas. Hoje já é possível comprar *Single Boards* específicas que já vêm com DD-WRT.

Para saber se há disponibilidade, vamos demonstrar. Deve-se entrar no site oficial do DD-WRT, no link https://dd-wrt.com/. Vamos precisar entrar na seção **Router Database** para buscar as versões compatíveis dos equipamentos. Caso tenha dúvidas, recorra ao Fórum. (Detalhes técnicos do EA6500 podem ser localizados no guia oficial da Linksys: https://downloads.linksys.com/downloads/userguide/1224699084559/EA6500*v2*UserGuide_En-FRCA.pdf.)

Ao procurar pelo modelo EA6500, o site lista automaticamente as versões compatíveis — isso mesmo, versões. Você talvez nunca tenha precisado saber disso, mas saiba agora: os routers possuem versão, e a versão é impactada pelo hardware interno, em específico o chipset. No caso acima temos duas versões, e para saber a versão tem que olhar o código de barras que está na caixa ou embaixo do aparelho, onde alguns possuem a identificação explícita da versão e outros não.

A instalação começa realizando o download no site; o nome da imagem pode mudar de acordo com a marca do seu router. Repare na figura a versão do chipset e no recorte do aparelho — talvez você precise abrir o equipamento para confirmar.

ATENÇÃO: Tive que fazer o download de uma versão muito antiga e depois atualizar. A versão que consegui instalar é a **1.1.27.144156**, que está neste link: https://drive.google.com/file/d/1aVNLwWKqfqfh9Xlzx9T9kV0nIYLiOfad/view?usp=drive_link.

Cada router é diferente; você terá que consultar o manual do seu para saber como se procede para trocar o **firmware**. Na imagem de um Cisco, por exemplo, encontramos um botão para upload. Essa é a pior parte: se cair a energia neste momento, pode causar problemas irreversíveis (o famoso "brick", quando o aparelho vira um tijolo). Tenha um nobreak ou um sistema de alimentação muito estável. A barra de progresso tem que chegar ao fim, e depois haverá uma instalação demorada — afinal, é uma memória flash sendo reescrita.

Vamos para o painel administrativo. Se tudo der certo, de forma muito lenta você será redirecionado para a página principal do DD-WRT. No meu caso particular, deste dispositivo particular, tive que instalar uma versão de 2013 e depois fazer um upgrade para 2025. Veja no topo a versão. Para mostrar a configuração de VPN, na aba **Services** clique sobre **VPN**. Ative a VPN e, logo em seguida, obtenha os dados de configuração. Cada operador de VPN tem uma interface e libera um arquivo diferente, então leia as documentações. Veja exemplos de configuração de VPN no DD-WRT (confirme a compatibilidade da sua VPN com o DD-WRT):

- **CyberGhost**: https://support.cyberghostvpn.com/hc/en-us/articles/213811885-Router-How-to-Set-Up-OpenVPN-on-DD-WRT-Routers
- **NordVPN**: https://support.nordvpn.com/hc/en-us/articles/20308623061265-DD-WRT-setup-with-NordVPN

De posse dos dados, é só configurar. Muito simples: salve e aplique as configurações. Para saber se deu certo, basta ir na aba **Status** e clicar na opção **OpenVPN** — veja se está `CONNECTED SUCCESS`. Se quer ter certeza, acesse o painel https://iplocation.net, mas certifique-se de que não está transmitindo a sua interface para lugar nenhum. Fisicamente eu não estou em Buenos Aires, mas virtualmente sim.

Conectando-se ao servidor VPN com o OpenVPN

A implementação mais conhecida que atua com VPN é o **OpenVPN**. Trata-se de um sistema que cria conexões seguras de ponta a ponta, entre o sistema cliente e o sistema servidor. Essa solução pode ser utilizada somente como cliente, somente como servidor, ou como cliente e servidor.

O OpenVPN permite que os pares (cliente e servidor) se autentiquem usando chaves secretas pré-compartilhadas, certificados ou usuário/senha. Quando utilizado em uma configuração multi cliente-servidor, permite que o servidor libere um certificado de autenticação para cada cliente, utilizando assinaturas e autoridade de certificação. Sempre recomendo saber como configurar um servidor VPN; isso é tratado no livro *Manual Debian GNU/Linux*. Aqui ficamos só no lado cliente.

O OpenVPN usa extensivamente a biblioteca de criptografia **OpenSSL**, bem como o protocolo **TLS**, e contém muitos recursos de segurança e controle. Ele usa um protocolo de segurança personalizado que utiliza SSL/TLS para a troca de chaves. O OpenVPN foi portado e incorporado em vários sistemas — por exemplo, o DD-WRT possui a função de servidor OpenVPN. O SoftEther VPN, um servidor VPN multiprotocolo, também possui uma implementação (própria) do protocolo OpenVPN.

OpenVPN ou WireGuard

Antes de entrar nos arquivos, uma nota de atualização que a fonte não cobre, mas que você precisa conhecer. O OpenVPN é o padrão consagrado, testado há mais de duas décadas, extremamente configurável e presente em praticamente qualquer lugar. Em compensação, é "pesado": muitas opções, código grande, negociação TLS custosa. Nos últimos anos surgiu o **WireGuard** como alternativa moderna. Ele é bem menor (poucos milhares de linhas de código, contra centenas de milhares do OpenVPN), roda no próprio **kernel** do Linux, usa criptografia fixa e moderna (sem a "sopa de letrinhas" de opções negociáveis), e por isso tende a ser mais rápido, com latência menor e reconexão quase instantânea ao trocar de rede — algo valioso para quem opera de celular ou muda de Wi-Fi.

O contraponto, do ponto de vista de privacidade, é que o WireGuard associa cada cliente a um IP interno fixo por padrão. Sem uma camada extra da operadora (o chamado *double NAT* ou tabelas dinâmicas), isso pode facilitar a correlação. Por isso os provedores sérios embrulham o WireGuard em soluções próprias (NordLynx e semelhantes). Regra prática: se a sua VPN oferece WireGuard e você quer velocidade, use; se precisa de máxima compatibilidade, atravessar firewalls chatos ou rodar dentro de um router antigo com DD-WRT, o OpenVPN ainda é o cavalo de batalha. Os conceitos de kill switch e de edição de rotas deste capítulo valem para **qualquer** um dos dois — o que muda é só o processo que sobe o túnel.

Os arquivos da conexão

Quando você contrata um serviço de VPN, é comum que o serviço entregue um arquivo compactado contendo 4 arquivos:

- Certificado do servidor — o arquivo `ca.crt`;
- Certificado do cliente, individual — o arquivo `client.crt`;
- Chave privada do cliente, individual — o arquivo `client.key`;
- Arquivo com toda a configuração necessária — o arquivo `openvpn.ovpn`.

(Pode acontecer de o servidor injetar o conteúdo de `ca.crt`, `client.crt` e `client.key` dentro de **tags** no próprio arquivo `openvpn.ovpn`, entregando tudo em um só arquivo.)

Além desses, costuma existir um arquivo chamado `pass.txt`. Ele fica nesse mesmo diretório porque é usado para armazenar usuário e senha do serviço de VPN. Esse arquivo é criado pelo próprio usuário e **não** vem dentro do compactado. Ele tem, na primeira linha, o *username* da VPN e, na segunda linha, a senha do serviço. Na listagem abaixo temos um arquivo `.ovpn` básico, para exemplificação:

```
client
dev tun
proto udp
remote 192.168.0.16 1194
tls-client
resolv-retry infinite
```

```
nobind
persist-key
persist-tun

ca ca.crt
cert client.crt
key client.key
```

Vamos ler esse arquivo com calma. A diretiva `client` indica ao OpenVPN que este arquivo é usado para iniciar a ponta do cliente e que ele irá se conectar a um serviço. Quando o OpenVPN iniciar, ele criará uma interface de rede virtual, cujo nome deve começar com `tun` e ter um número — se for o único túnel, será `tun0`. A diretiva `proto udp` indica que o protocolo de camada de transporte será o **UDP**. Há a possibilidade de uso de **TCP**, mas isso normalmente não é viável (rodar um túnel confiável de TCP por cima de outro TCP degrada muito o desempenho — o chamado *TCP meltdown*).

Na linha `remote 192.168.0.16 1194` temos o IP e a porta do serviço VPN. Nem sempre é essa porta; é comum haver várias portas em um servidor. Também é comum ter muitas entradas com IPs e portas diferentes — neste arquivo temos apenas 1 entrada. As diretivas de `tls-client` até `persist-tun` são parâmetros que variam muito de servidor para servidor e de cliente para cliente. Inclusive, é comum o servidor mandar parâmetros que não existem mais; isso pode variar. (`persist-key` e `persist-tun` mantêm chave e interface vivas entre reconexões; `nobind` deixa o sistema escolher a porta local; `resolv-retry infinite` fica tentando resolver o nome indefinidamente.)

Já nas linhas `ca ca.crt`, `cert client.crt` e `key client.key` temos os endereços dos certificados — e aqui tome cuidado. Veja que o endereço do arquivo é **relativo**, então deve-se navegar até o diretório com o comando `cd` e, de lá, executar o comando `openvpn`. Eu prefiro alterar essas três linhas do arquivo com endereços reais e completos, para evitar isso. Assim, posso executar o comando `openvpn` de qualquer ponto do sistema de arquivos. Para iniciar a VPN levando em consideração os arquivos acima, execute os comandos no terminal:

```
cd /var/kfm/vpn
sudo /usr/sbin/openvpn --config ./openvpn.ovpn --auth-user-pass ./pass.txt
```

Como, ao executar o comando `cd`, o cursor do sistema de arquivos se moveu para `/var/kfm/vpn`, e todos os arquivos estão lá, é só executar o `openvpn` passando o arquivo de configuração e o arquivo com usuário e senha. Lembrando que o arquivo de usuário e senha tem duas linhas: na primeira o usuário, na segunda o *password*.

Vazamento de DNS também acontece em VPN

Vale reforçar um ponto que muita gente pensa ser exclusivo do Tor: o **vazamento de DNS (DNS leak)** também acontece em VPN, e é sorrateiro. Antes de acessar um site, sua máquina traduz o nome (por exemplo, `google.com`) em um IP, usando o protocolo **DNS**. Se essa consulta sair pelo resolvedor do seu provedor "por fora" do túnel — algo comum quando o sistema mantém o servidor DNS antigo configurado —, então mesmo com todo o tráfego web dentro da VPN, a **lista do que você quis acessar** fica registrada nos logs do DNS do seu ISP. O IP muda, o rastro do DNS permanece.

Para uma VPN, o cuidado é garantir que o `openvpn.ovpn` empurre o DNS do provedor de VPN e que o sistema realmente o use. A validação é a mesma do capítulo do Tor: use `nslookup` e confirme quem responde.

```
nslookup exemplo.com
```

```
# Server: 10.x.x.x (DNS da VPN)  → OK
# Server: 192.168.0.1 (seu roteador) → PERIGO, DNS vazando para o ISP
```

E, de novo, a conclusão de fundo do livro se aplica: DNS seguro deveria ser responsabilidade do **sistema operacional**, não de cada aplicativo — só assim vale para todos os processos, e não apenas para as abas de um navegador.

Monitorando o túnel tunX

Caso opte por ter a VPN instalada diretamente na máquina, recomendo o projeto **KFishmonger**. Além de uma ferramenta de aleatoriedade de VPNs, ele tem a auto-conexão e uma interface de monitoramento direto na área de trabalho. Um recorte de *print-screen* da área de trabalho mostra a VPN em uso e coerente com a interface de rede — ou seja, o IP que aparece no monitor bate com a interface `tunX` ativa, e não com a interface física.

Esse monitoramento constante não é firula. Como veremos a seguir, o modelo de ameaça mais perigoso de uma VPN é ela cair **sem você perceber**. Um indicador sempre visível na tela — mostrando o país de saída e a interface ativa — é a primeira linha de defesa: se o número mudar para o seu país real, você vê na hora.

Para instalar é muito simples. Na página inicial do projeto você encontra um arquivo chamado `install.sh`, compatível com Debian, Ubuntu, Kali e outras distros baseadas em Debian. Mas veja os códigos-fonte antes de instalar — o projeto está em https://github.com/naoimportaweb/kfishmonger.

```
git clone https://github.com/naoimportaweb/kfishmonger
cd kfishmonger
# leia os fontes ANTES de rodar
sudo ./install.sh
```

VPN caindo sem aviso: o problema que justifica o kill switch

Um único erro pode entregar o hacker. Veja o caso de **Hector Xavier Monsegur** (o "Sabu", do LulzSec), que só foi localizado e preso porque, em uma única vez, esqueceu de ligar o Tor. Uma vez. Foi o suficiente para o FBI cravar o seu IP real, e a partir daí ele foi virado e passou a colaborar contra os próprios companheiros.

As VPNs **não foram criadas para o anonimato** — nesse aspecto elas são piores que o Tor. E há um motivo técnico específico que as torna traiçoeiras: a VPN pode parar de funcionar e o hacker **nem percebe**. Tudo continua funcionando normalmente, porque o tráfego volta para o ISP muito rápido, de forma transparente. O navegador não trava, o site abre, o download continua — e o seu IP real está escancarado. Esse é o modelo de ameaça que justifica tudo o que vem a seguir: não é que a VPN vá ser "hackeada"; é que ela vai simplesmente **cair** (queda do processo, timeout, troca de servidor, suspensão do notebook) e deixar você exposto no exato momento em que você acha que está protegido.

Para entender a solução, precisamos entender **rotas**. Na figura a seguir, repare nas rotas antes da VPN: a rota `default` aponta para o router de saída do ISP. Ou seja, tudo que não tem regra específica vai por ali, direto para o seu provedor.

Quando uma VPN é ligada, as rotas são modificadas. Após ligar a VPN, são introduzidas as regras necessárias para o túnel, principalmente a rota que desvia todas as requisições para a `tun0` (o túnel). Os protocolos capturados pelo túnel são enviados então para o IP do serviço

VPN (por exemplo, `185.177.125.173`), que naturalmente segue para o gateway da rede, mas agora **criptografados e tunelados** — estamos seguros.

O ponto crítico está no que acontece quando o túnel cai. Desligando a VPN (parando o processo `openvpn`), sobram somente as regras que existiam antes: voltamos para o ISP, **sem tunelamento**, e sem nenhum aviso. O sistema simplesmente volta a usar a rota `default` para o gateway do provedor. É exatamente essa "volta silenciosa" que precisamos impedir. Para isso temos duas soluções:

1. Um script que fica de segundo a segundo validando a VPN, chamado de **Kill Switch**;
2. Alterar as regras de roteamento para somente enviar dados através do tunelamento.

Kill Switch por script

A solução do script Kill Switch pode ter um problema: por um pequeno intervalo de tempo, o script está processando e investigando, para só atuar quando perceber a falha — e esse "quando perceber" pode ser tarde. A vantagem é que dá para testar além da simples presença da VPN, tal como **qual país** está saindo a conexão. Isso é poderoso: em vez de só verificar "a VPN caiu?", você verifica "a minha conexão está saindo por um país que eu não quero?".

Crie um arquivo chamado `killswitch.py` com um editor de scripts e digite o código abaixo:

```
#!/usr/bin/python3
import json, requests, subprocess, time
def get_ip():
    r = requests.get("https://wtfismyip.com/json")
    return json.loads(r.text)
def main():
    TEMPO_SEGUNDOS_AGUARDAR = 30
    country = ["brazil", "panama"]
    while True:
        try:
            retorno = get_ip()
            if retorno["YourFuckingCountry"].lower() in country:
                subprocess.call(["ip", "route", "del", "default"])
                print("Parando a Internet, IP do " + retorno["YourFuckingCountry"] + "
detectado.")
                TEMPO_SEGUNDOS_AGUARDAR = 0
                break
        except:
            print("Continue....")
        finally:
            time.sleep(TEMPO_SEGUNDOS_AGUARDAR)
if __name__ == "__main__":
    main()
```

Basicamente, caso o IP seja de um determinado país, o script elimina a regra `default` do Linux, impedindo qualquer conexão. Repare na lógica: a lista `country` contém os países que **não** podem sair — no exemplo, `brazil` (o país real do operador) e `panama`. A cada 30 segundos, o script consulta o serviço `wtfismyip.com`, que devolve em JSON o país de saída da conexão. Se o país de saída estiver na lista proibida, significa que a VPN caiu (voltou para o Brasil) ou que você está saindo por um país indesejado — e o script derruba a rota default

na hora, cortando toda a Internet. Essa abordagem é ótima para quem utiliza o Tor Browser em cima.

O preço a pagar: depois disso, nada mais se conectará à Internet. Então, após resolver o problema (subir a VPN de novo), deve-se adicionar novamente a regra default. Neste exemplo, o IP `10.0.2.2` é o IP do router gateway:

```
1 sudo ip route add default via 10.0.2.2
```

Vale ser honesto sobre a fragilidade dessa abordagem, para você decidir com consciência. Entre uma verificação e outra existe uma **janela** (aqui, até 30 segundos) em que a VPN pode ter caído e o tráfego pode estar vazando pelo ISP sem que o script tenha percebido ainda. Diminuir o intervalo reduz a janela, mas aumenta o número de requisições ao serviço externo (que pode te bloquear por excesso). É um Kill Switch **reativo**: ele conserta depois que o problema aconteceu. A solução da próxima seção é **preventiva** — ela torna o vazamento impossível por construção, em vez de detectá-lo depois.

Editando as rotas: o kill switch que não tem janela

Todo arquivo `.ovpn` tem um atributo chamado `remote`; nesse atributo temos um IP de servidor ou um domínio de servidor. A solução é um pouco custosa quanto à digitação, mas é fácil de entender. Primeiro você deve ler o arquivo `.ovpn` e obter o IP dos servidores VPN aos quais irá se conectar.

É mais fácil para as empresas criarem arquivos com **domínios** em vez de IP explícito, pois assim podem trocar de servidores com facilidade. Se colocarem o IP direto e precisarem trocar o servidor, podem ter dificuldades com os clientes (chamados de suporte). Se a sua empresa de VPN não usa IPs explícitos, então você terá que traduzir o domínio com `nslookup`:

```
1 nslookup servidor.suavpn.com
```

A ideia central é inverter a lógica do roteamento. Em vez de "mande tudo para a Internet, exceto o que a VPN capturar", passamos a dizer "**não mande nada para a Internet, exceto para o IP do servidor VPN**". Assim, se a VPN cair, não sobra nenhuma rota que leve ao ISP — o vazamento se torna fisicamente impossível, sem janela de tempo nenhuma.

A primeira coisa a fazer é remover a regra `default` que aponta para o router gateway (o que o liga ao ISP). Depois, para cada IP do serviço de VPN, cria-se uma regra específica apontando para o router gateway. Vou tomar como exemplo um arquivo `.ovpn` que tem como `remote` o servidor `185.177.125.173`, e o router gateway no IP `10.0.2.2`. Temos então as seguintes regras:

```
  1 sudo ip route del default
104 sudo ip route add 185.177.125.173 via 10.0.2.2
```

Ao observar as regras resultantes, só há **uma** forma de enviar dados para a **WAN**: conectando-se ao IP `185.177.125.173` da empresa de VPN. Então a VPN irá funcionar normalmente. Se tentar abrir qualquer aplicativo, verá que nada funciona (nada que precise da Internet), porque não há rota para mais nada. A partir daí, podemos iniciar a VPN conforme o início deste tópico, e as rotas do túnel serão criadas para atender aos requisitos da empresa de VPN. Se executar o envio de **ICMP** com um comando `ping`, verá que tudo irá funcionar.

E aqui está a mágica: se desligar a VPN, o envio de ICMP para o resto da Internet **para** — nada vai passar para o ISP, e você sempre estará seguro. Não existe a janela de 30 segundos do script; a proteção é estrutural. O custo é a rigidez: se a VPN trocar de IP de servidor (comum quando o `remote` é um domínio com vários IPs), você precisa reescrever a regra para o novo IP, senão fica sem conexão. Para esse exemplo, desenvolvi um script que realiza as alterações nas rotas automaticamente, lendo o `.ovpn` e montando as regras. Esse script Python está acessível em https://github.com/naoimportaweb/avulso/blob/main/blockdefault.py (o `killswitch.py` da seção anterior está no mesmo repositório: https://github.com/naoimportaweb/avulso/blob/main/killswitch.py).

Quando uma VPN não basta: multi-hop e Tor por cima

Para fechar, um conceito que eleva o jogo. Como o ponto fraco da VPN é a confiança concentrada num único provedor, uma técnica é o **multi-hop** (também chamado de *double VPN*): o tráfego passa por **dois** servidores da VPN, em países diferentes, antes de sair. O primeiro servidor conhece o seu IP real mas não o destino; o segundo conhece o destino mas não o seu IP — uma ideia emprestada do próprio Tor, que distribui a confiança. Não é tão forte quanto três relays independentes operados por gente diferente, porque no multi-hop os dois saltos costumam ser da **mesma empresa** (mesma confiança concentrada, só que em duas caixas), mas quebra a correlação trivial de um salto só.

A combinação mais séria, quando o alvo é anonimato de verdade e não só privacidade, é rodar o **Tor por cima da VPN**: você conecta a VPN primeiro (o ISP só vê que você fala com a VPN, nem sabe que você usa Tor) e, por dentro dela, sobe o Tor (o nó de guarda vê o IP da VPN, não o seu). Aí você soma o que cada ferramenta faz de melhor: a VPN esconde do provedor o fato de você usar Tor, e o Tor devolve o anonimato distribuído que a VPN sozinha nunca entrega. Guarde a distinção com que abrimos o capítulo: **VPN é privacidade, Tor é anonimato** — e quem entende os dois sabe quando usar cada um, e quando empilhá-los.

Ferramentas citadas

- **OpenVPN** — implementação cliente/servidor de VPN sobre SSL/TLS (OpenSSL), padrão da indústria, altamente configurável; **código aberto** (licença GPLv2).
- **WireGuard** — VPN moderna embutida no kernel Linux, com criptografia fixa e foco em desempenho e simplicidade; **código aberto** (licença GPLv2).
- **DD-WRT** — firmware Linux para routers, com cliente OpenVPN embutido, permitindo o túnel no nível do roteador; **código aberto** (baseado em GPL; alguns módulos com restrições de redistribuição).
- **SoftEther VPN** — servidor VPN multiprotocolo (inclui implementação própria do OpenVPN); **código aberto** (licença Apache 2.0).
- **KFishmonger** — projeto próprio para automação, aleatorização e monitoramento de VPNs em Debian/Kali; **código aberto** (https://github.com/naoimportaweb/kfishmonger).
- **killswitch.py / blockdefault.py** — scripts próprios de kill switch (por verificação de país e por edição de rotas); **código aberto** (https://github.com/naoimportaweb/avulso).
- **wtfismyip.com** — serviço web que retorna IP e país de saída em JSON, usado pelo kill switch; serviço público gratuito.

Referências

- YONAN, James. **OpenVPN — Security Overview** e documentação oficial. OpenVPN Technologies. Disponível em: https://openvpn.net/
- DONENFELD, Jason A. **WireGuard: Next Generation Kernel Network Tunnel**. NDSS Symposium, 2017.
- DD-WRT PROJECT. **Router Database e OpenVPN Client**. Disponível em: https://dd-wrt.com/
- OLSON, Parmy. **We Are Anonymous: Inside the Hacker World of LulzSec, Anonymous and the Global Cyber Insurgency**. Little, Brown and Company, 2012. (Caso Hector Xavier Monsegur / Sabu.)
- ELETRONIC FRONTIER FOUNDATION. **Choosing the VPN That's Right for You** e materiais sobre limites de anonimato de VPNs. Disponível em: https://ssd.eff.org/

Rede I2P

A rede I2P é uma rede distribuída de nós participantes, e esta frase vai definir tudo o que vou explicar neste texto. A rede Tor é uma rede descentralizada, e há uma grande diferença entre ser distribuída e descentralizada — essa distinção é o coração deste capítulo. Mas antes de iniciar toda a teoria, tenho que dizer que vou dedicar este tópico a dois grandes amigos, o companheiro Petrov (@Petrovrz) — um cara que sabe muito e me enche o saco para usar I2P — e Jhon China Team (@Wkskrrt), um colega do Telegram que sempre esteve ativo e focado em anonimato. É para vocês, meninos.

Centralizada, descentralizada, distribuída

Uma rede centralizada, como o WhatsApp, possui servidores centrais que são obrigatoriamente registrados no nome da empresa para que sejam auditados. Mas essas auditorias não são feitas por pessoas comuns: são feitas por órgãos repressores. E digo mais — o WhatsApp e o Telegram não querem a censura, mas, para que seus modelos de negócio sobrevivam, dependem de aprovações totalitárias de máfias locais, vulgarmente conhecidas como governos.

Esses servidores centralizados são facilmente alcançados por governos, que, com seus agentes, podem espionar pessoas ou até mesmo interagir com elas da forma mais desumana possível — que são as ações de prisão e assassinato. Agora, e se fosse possível ter uma rede descentralizada de servidores ocultos? Modelos de negócio como WhatsApp e Telegram não sobreviveriam.

Em uma estrutura descentralizada, os nós (servidores) podem ser mantidos por entusiastas ou até mesmo por empresas e governos — e esta é a grande falha da rede Tor. Em toda a sua história, o Tor foi concebido por uma máfia que pretendia criar um ambiente seguro para seus assassinos, que estavam em territórios de outra máfia (os governos). É natural que a exposição de assassinos não seja uma coisa legal, então uma rede anônima se torna necessária. Vale registrar o fato técnico por trás dessa provocação: o roteamento **Onion** nasceu no Laboratório de Pesquisa Naval dos Estados Unidos e o Tor recebeu por muito tempo financiamento do próprio governo americano.

A rede Tor precisava de alguns ingredientes:

- Uma criptografia forte e bem arquitetada;
- Uma arquitetura de nós descentralizada, para poder rotear a mensagem por vários servidores (nós);
- Uma grande quantidade de pessoas usando, para dificultar a segregação das mensagens de assassinos (do governo) das mensagens de pessoas comuns.

Uma rede Tor é um conjunto de servidores espalhados pelo planeta, dos quais muitos são mantidos por governos, outra grande parcela por pessoas entusiastas e uma parcela menor por empresas. E, para piorar, mais de 90% dos servidores estão em países controlados pelos 14 olhos.

Mas se todos os nós fossem os próprios usuários, então cada usuário teria um **PIPE** de conexão entre usuários nesta rede — isso seria uma rede distribuída. E, entre centralizada, descentralizada e distribuída, aquela que é pior para se combater é a distribuída. O caminho entre usuários é dado pela comunicação, ou rota, entre eles, e assim é possível ocultar serviços nesta rede distribuída. A diferença é estrutural: numa rede descentralizada ainda

existem nós "graúdos" que concentram função e podem ser capturados; numa rede verdadeiramente distribuída não há hierarquia — cada participante é, ao mesmo tempo, cliente, servidor e roteador dos outros, e não sobra um ponto único onde a repressão possa apertar.

Tor e I2P não fazem a mesma coisa

Uma rede Tor disponibiliza apenas o serviço de roteamento, tanto para a rede de serviços da Internet quanto para uma rede Tor local, conhecida como Onion. Já um usuário que utiliza I2P consome serviços sobre protocolos HTTP/HTTPS, SMTP, SSH, SMB, PROXY, etc. — ou seja, na I2P encontramos **serviços**, e um desses serviços chama-se HTTP/HTTPS PROXY. Outra grande diferença é que na rede Tor trabalha-se a conexão **socket**, enquanto na rede I2P trabalha-se com serviços de mensagens.

Convém, então, deixar a comparação direta, porque as duas ferramentas resolvem problemas diferentes e não competem de verdade. O **Tor** é melhor quando o objetivo é acessar a *surface web* de forma anônima (saindo por um nó de saída para a Internet comum) e para consumir serviços `.onion` publicados por terceiros — ele foi otimizado para o cliente que sai da rede. A **I2P**, ao contrário, é melhor para serviços internos e para comunicação P2P *dentro* da própria rede: ela foi desenhada de dentro para dentro, não de dentro para fora. Onde o Tor pensa em "circuito bidirecional para um destino", a I2P pensa em "cada aplicação publica seu próprio destino oculto e conversa com outros destinos ocultos". Por isso a I2P brilha em mensageria, torrent anônimo e *eepsites*, e é desajeitada para simplesmente "navegar na web normal".

Há ainda uma diferença de fundo no jeito de embrulhar os pacotes. O Tor usa **onion routing**, em que cada mensagem viaja sozinha, embrulhada em camadas. A I2P usa **garlic routing** (roteamento em alho), e o nome não é gratuito: um "dente" isolado seria um *onion*, mas o "alho" agrupa **vários dentes** — isto é, várias mensagens de destinos e finalidades diferentes — dentro de um único pacote criptografado. Agrupar mensagens de origens distintas no mesmo envelope dificulta ainda mais a análise de tráfego, porque o observador não consegue nem contar quantas conversas independentes passaram por ali.

Serviços ocultos, eepsites e o problema dos nomes

O uso da I2P normalmente se dá para aplicações internas da rede, os chamados **Hidden Services**. Esses Hidden Services podem ser qualquer serviço de troca de mensagem e, principalmente, websites — os chamados **eepsites**. Outros serviços muito utilizados são o **I2PSnark**, que é um BitTorrent oculto, e o **I2PTunnel**, para aplicações como IRC e SSH (nunca use SSH).

Não vou me aprofundar no uso da *surface web* por meio do proxy I2P, pois vejo a rede I2P como uma rede anônima para acesso a recursos ocultos. Mas antes de demonstrar o uso de serviços na rede oculta da I2P, é preciso dizer como a comunicação é tratada nela. Em primeiro lugar, a rede I2P não possui um sistema de resolução de nomes como o **DNS** da *surface web*. O que existe é um sistema básico de nomes num arquivo do tipo texto, no qual se vincula o serviço a um domínio genérico com terminação `.i2p`. Estes endereços podem ser em **base 32** ou **base 64**, o que dificulta muito a ação humana de decorá-los — por isso é comum manter uma agenda de endereços em arquivo texto. Veja este exemplo de URL:

```
1 http://udhdrtrcetjm5sxzskjyr5ztpeszydbh4dpl3pl4utgqqw2v4jna.b32.i2p
```

Isso é fundamental. Veja: muitos hackers já se entregaram porque não conseguem criar nomes ou *nicknames* disruptivos em sua mente e sempre seguem padrões de nomes. Então

uma URL complexa e definida pela plataforma é fundamental para forçar o hacker a se manter anônimo.

Na I2P não existe um sistema centralizado para resolver um destino como o correspondente a um `.i2p`. O que existe é um sistema de nomeamento cuja gestão pode ser feita por meio da aplicação **SusiDNS**, que se encontra instalada por padrão em todos os roteadores da rede I2P e pode ser acessada pela URL `http://127.0.0.1:7657/susidns/index`. Mas saiba que um serviço oculto na rede I2P pode apontar para qualquer tipo de servidor e serviço web, até serviços SSH, FTP, SMB, POP, SMTP, etc. Na rede I2P não existe restrição de protocolos, e você pode criar qualquer tipo de protocolo ou serviço; conforme já dito, os mais conhecidos e usados são os eepsites, afinal o serviço mais consumido é o de websites.

Quando o serviço I2P não é capaz de resolver o destino de um serviço oculto, uma interface web é exibida dizendo que houve uma falha ao resolver o domínio. Uma possibilidade é mapear o serviço oculto com o endereço base32 ou base64, criando o seu próprio **addressbook**. E, mesmo assim, deve-se esperar um tempo para poder usá-lo — na rede I2P as coisas não são tão dinâmicas quanto na *surface*.

Por baixo desse sistema de nomes existe a peça que faz a rede distribuída funcionar sem um servidor central: a **netDb** (*network database*). Ela é um banco de dados distribuído que guarda dois tipos de registro — os **RouterInfos** (como falar com cada roteador da rede) e os **LeaseSets** (como chegar a cada destino/serviço oculto). Esse banco não fica num lugar; é espalhado entre os próprios roteadores usando uma variante da tabela de espalhamento distribuída **Kademlia (DHT)**. É a netDb que o SusiDNS e o addressbook consultam por baixo do pano quando você digita um `.i2p`, e é por ela que um novo roteador "aprende" a rede ao entrar. Como essa consulta é distribuída e leva tempo para propagar, aqui está uma das razões técnicas de a I2P ser lenta e "não dinâmica".

Arquitetura: túneis unidirecionais

Devo começar descrevendo que esta arquitetura é distribuída, e realmente distribuída; para reafirmar, os nós participantes formam uma rede não hierarquizada. A I2P cria um esquema de **túneis**, e esse é o componente central de sua arquitetura: um túnel é um conjunto de roteadores que se encarregam do envio dos pacotes de dados até um destino, permitindo uma comunicação impossível de se rastrear na rede I2P.

Quando um usuário inicia seu **I2P router**, ele automaticamente vira um roteador nesta imensa rede — e, inclusive, a velocidade desta rede é decorrência do número de usuários e dos serviços utilizados por eles. Não espere ver vídeos no YouTube de forma anônima por aqui, muito menos baixar enormes arquivos, pois a velocidade é lastimável. Porém, não se pode negar que é a forma mais próxima da perfeição quando o quesito é anonimato. A razão dessa proximidade com a perfeição é justamente arquitetural: como todo usuário é nó, todo tráfego é P2P e o *garlic routing* mistura mensagens de vários remetentes, não sobra a assimetria "cliente que só consome / servidor que só serve" que o Tor tem no nó de saída — e é essa mesma característica que cobra o preço da lentidão.

Acredito que a comunidade deva se focar em serviços de comunicação e resiliência à censura; a questão dos grandes downloads apenas estraga a performance da rede por um valor social ridículo. Bom, vocês sabem que sou chato e só estudo, trabalho e troco mensagens anárquicas com outros *players*.

Como todo usuário vira um *router* para roteamento de pacotes, é natural que ele também passe a ser rota. Uma característica importante, que a meu ver é única, é que os túneis

possuem funcionamento em um único sentido. Para isso, cada *router* cria um **túnel de entrada** e um **túnel de saída**: cada saída é marcada por um **gateway** e cada entrada por um **endpoint**. Essa unidirecionalidade é uma diferença marcante em relação ao Tor, cujo circuito é bidirecional. Na I2P, para uma simples troca de "pergunta e resposta" entre dois destinos, chegam a estar envolvidos quatro túneis distintos: o de saída de quem pergunta, o de entrada de quem responde, e o par simétrico para o caminho de volta.

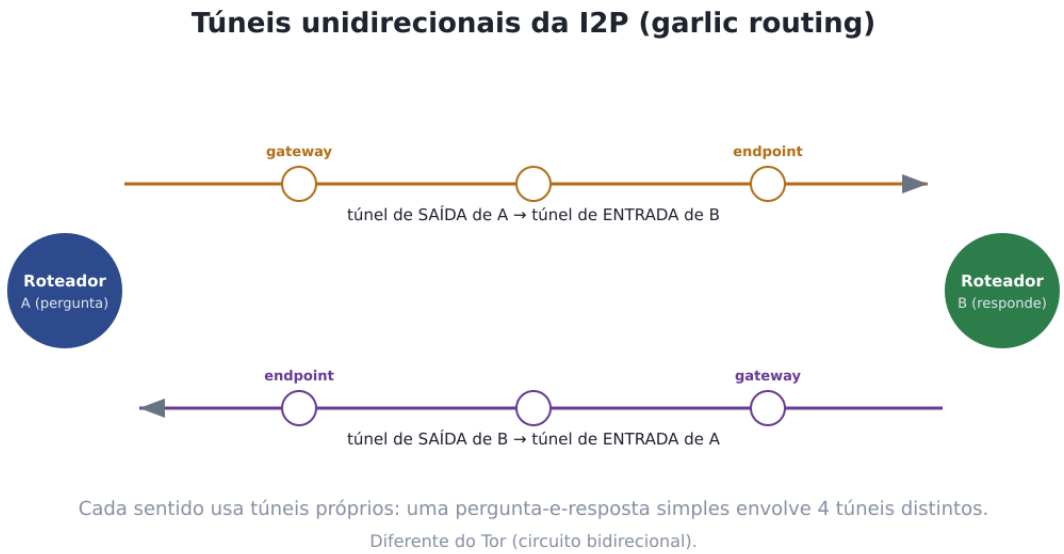

Figura: Túneis unidirecionais da I2P.

Os quatro processos do túnel e as mensagens I2NP

Depois de um túnel ser construído, as mensagens **I2NP** (*I2P Network Protocol*) são processadas e passadas através dele. A operação do túnel tem quatro processos distintos, assumidos por vários pares ao longo do caminho:

1. Primeiro, o *gateway* do túnel acumula um número de mensagens I2NP e as pré-processa em mensagens de túnel para entrega;
2. Em seguida, esse *gateway* criptografa os dados pré-processados e os encaminha para o primeiro salto;
3. Aquele par, e os pares participantes subsequentes do túnel, descompactam uma camada da criptografia, verificam se a mensagem não é uma duplicata e, então, a encaminham para o próximo par;
4. Eventualmente, as mensagens do túnel chegam ao ponto final (*endpoint*), onde as mensagens I2NP originalmente empacotadas pelo *gateway* são remontadas e encaminhadas conforme solicitado.

Os participantes intermediários do túnel não sabem se estão em um túnel de entrada ou de saída; eles sempre "criptografam" para o próximo salto. Portanto, aproveita-se a criptografia **AES simétrica** para "descriptografar" no *gateway* do túnel de saída, de modo que o texto simples só seja revelado no ponto de extremidade de saída.

A I2P é uma rede inerentemente comutada por pacotes, mesmo com estes túneis, o que lhe permite tirar proveito de vários túneis rodando em paralelo, aumentando a resiliência e balanceando a carga. Fora da camada I2P central, há uma biblioteca de *streaming* de ponta a ponta opcional disponível para as aplicações cliente, expondo a operação de TCP — incluindo reordenação de mensagens, retransmissão, controle de congestionamento, etc.

Mensagens de tamanho fixo e criptografia em camadas

A função de um *gateway* de túnel é fragmentar e empacotar as mensagens I2NP em tamanho fixo e criptografar as mensagens do túnel. As mensagens do túnel contêm os seguintes campos:

- Um **ID de túnel** de 4 bytes;
- Um **IV** (vetor de inicialização) de 16 bytes;
- Um **checksum** para verificação;
- **Enchimento** (*padding*);
- Uma ou mais instruções de entrega.

Os IDs de túnel são números de 4 bytes usados em cada salto; a partir deles, os participantes sabem qual ID devem escutar para receber mensagens. Os túneis em si são de curta duração, em torno de **10 minutos**. Mesmo que os túneis subsequentes sejam construídos usando a mesma sequência de pares, o ID do túnel de cada salto mudará — o que impede que um observador acompanhe um mesmo túnel por muito tempo.

Para evitar que adversários marquem as mensagens ao longo do caminho ajustando o tamanho da mensagem, todas as mensagens de túnel têm **1024 bytes fixos** de tamanho. Para acomodar mensagens I2NP maiores, bem como para suportar as menores de forma mais eficiente, o *gateway* divide as mensagens I2NP maiores em fragmentos contidos dentro de cada mensagem de túnel. O *endpoint* tentará reconstruir a mensagem I2NP a partir dos fragmentos por um curto período de tempo, mas irá descartá-los conforme necessário.

Quando um par recebe uma mensagem de túnel, ele verifica se ela veio do mesmo salto anterior de antes. Se o par anterior for um roteador diferente, ou se a mensagem já tiver sido vista, ela é descartada. O participante então criptografa o que recebeu com **AES256/ECB**, usando sua chave e um IV, e encaminha a tupla `{nextTunnelId, nextIV, encriptadoData}` para o próximo salto. Essa dupla criptografia do IV (antes e depois do uso) ajuda a endereçar uma determinada classe de ataques de confirmação.

Após receber e validar uma mensagem de túnel no último salto, a forma como o *endpoint* recupera os dados codificados pelo *gateway* depende de o túnel ser de entrada ou de saída. Para túneis de saída, o *endpoint* criptografa a mensagem com sua chave de camada, assim como qualquer outro participante, expondo os dados pré-processados. Para túneis de entrada, o ponto final também é o criador do túnel, então ele pode simplesmente descriptografar iterativamente o IV e a mensagem, usando as chaves de camada e de IV de cada etapa na ordem inversa.

Neste ponto, o ponto de extremidade do túnel tem os dados pré-processados enviados pelo *gateway*, que ele pode então analisar para extrair as mensagens I2NP incluídas e encaminhá-las conforme solicitado nas suas instruções de entrega.

Ferramentas citadas

- **I2P (Invisible Internet Project)** — rede anônima distribuída, com roteamento *garlic* e túneis unidirecionais; o roteador de referência é o **I2P Java Router**. Licença **código aberto** (majoritariamente domínio público / BSD / MIT, conforme o componente).
- **I2PSnark** — cliente BitTorrent embutido no roteador I2P para compartilhamento de arquivos anônimo dentro da rede; **código aberto**.

- **I2PTunnel** — ferramenta do roteador para expor e consumir serviços TCP genéricos (IRC, HTTP, etc.) como serviços ocultos; **código aberto**.
- **SusiDNS** — aplicação web de gestão de *addressbook* e nomes `.i2p` embutida no roteador; **código aberto**.

Referências

- THE I2P PROJECT. **I2P: A Scalable Framework for Anonymous Communication** (documentação técnica). Disponível em: https://geti2p.net/en/docs
- THE I2P PROJECT. **Tunnel Routing** e **Garlic Routing and "Garlic" Terminology**. Disponível em: https://geti2p.net/en/docs/how/tunnel-routing
- THE I2P PROJECT. **The Network Database (netDb)**. Disponível em: https://geti2p.net/en/docs/how/network-database
- THE I2P PROJECT. **Comparing I2P to Tor**. Disponível em: https://geti2p.net/en/comparison/tor

Browsers e Telemetria

Por definição, um navegador é um **sandbox** (caixa de areia) e um **ponto final** (endpoint): ele executa código de terceiros — todo site é código de terceiros — dentro de uma cerca que deveria impedir esse código de tocar o resto do computador. Mas, como o navegador precisa atender pessoas leigas que lutam por mais facilidades e menos segurança, ele deixou de ser só uma cerca e passou a ser, também, um meio de insegurança. Há uma briga feroz sobre qual navegador é o melhor, e sempre que sai uma vulnerabilidade um grupo se vangloria enquanto o outro chora. Mas saiba: todo ano sai de uma a duas vulnerabilidades que levam à evasão da sandbox e à execução de código malicioso por script no computador do usuário, e isso sempre aconteceu, está acontecendo e sempre acontecerá, em todos os navegadores. Este capítulo trata de duas coisas que a maioria dos usuários nunca vê: de que material os navegadores são feitos, e o que eles falam pelas suas costas — a **telemetria**.

A base dos navegadores e a abertura do projeto

Boa parte dos navegadores estende de um projeto chamado **Chromium**. O Chromium é um projeto de código aberto mantido principalmente pela Google e serve de base para o Google Chrome, o Brave, o Microsoft Edge, o Opera e outros. Muitas vulnerabilidades no Chromium afetam uma grande quantidade de navegadores ao mesmo tempo, e por isso há uma certa desconfiança da comunidade hacker sobre navegadores que usam o Chromium como base. Veja: é um alvo óbvio pelo tamanho do impacto. Quando um bug de execução remota aparece no motor do Chromium, ele não afeta um navegador, afeta uma família inteira — e essa concentração tem nome.

O que os biólogos chamam de **monocultura (monoculture)** vale para software: quando quase todo mundo planta a mesma variedade, uma única praga derruba a lavoura toda. Hoje o Chromium (e o motor de renderização Blink, que veio dele) domina a web a tal ponto que muitos sites passaram a ser testados só nele, e uma falha no motor vira uma falha planetária de uma tacada só. É o oposto da diversidade que torna um ecossistema resiliente. Por isso, do ponto de vista de defesa, manter motores alternativos vivos não é romantismo: é reduzir a superfície comum de ataque. Guarde esse raciocínio, porque ele explica boa parte das minhas ressalvas ao longo do capítulo.

O outro grande motor nasceu do projeto **Gecko**, que teve como origem o Netscape. No começo ele era lento e problemático, principalmente em relação aos padrões do **W3C** (o consórcio que define os padrões da web). Com o tempo esse motor de execução foi melhorado e passou a ser peça fundamental do **Firefox**, um navegador muito robusto e bem aclamado no mundo hacker — mas o autor deste livro tem ressalvas, e vou detalhá-las adiante.[^1] Existe ainda um terceiro motor relevante, o **WebKit**, que a Apple usa no Safari e que aparece mais à frente quando eu chegar ao Epiphany. Na prática, então, a web inteira roda sobre basicamente três motores: Blink/Chromium, Gecko e WebKit — e saber em qual motor um navegador se apoia já diz muito sobre os riscos que ele herda.

[^1]: Foi notado em 2024 um excesso de dados de telemetria no navegador Mozilla Firefox — o que motiva a análise prática da seção de telemetria.

A tabela abaixo resume quatro navegadores populares mais o DuckDuckGo em quatro critérios básicos. Ela é uma foto de conjunto, não um veredito final — os testes das próximas seções é que vão dizer o que cada um faz na prática, sem intervenção humana.

A figura a seguir resume o comparativo entre os cinco navegadores.

Figura: Comparativo de recursos de privacidade entre navegadores.

Por mais ridículo que pareça, não é de hoje que os navegadores pensam mais em como vender dados de usuários do que em realmente ser um bom navegador — tanto que hoje muitos agem mais como uma agência de coleta de dados[^2] para anunciantes do que como ferramenta. Esses navegadores rastreiam e armazenam históricos de navegação, interesses e até o que digitamos. Naturalmente, vendem tudo isso como um produto para empresas de venda.

[^2]: Chamamos isso de **telemetria de navegador**.

Embora a navegação anônima ou privada pareça uma opção segura, ela ainda deixa você e seus dados expostos. Embora a navegação privada apague suas informações locais, seu endereço IP e sua localização continuam sendo compartilhados com todos os sites, anúncios e rastreadores carregados no navegador. Essas informações podem então ser vendidas a terceiros. O modo privado protege você do próximo usuário da sua máquina — não do site que você visita.

O Firefox sempre foi uma ótima opção de privacidade e segurança, mas o usuário precisa primeiro ajustar as configurações para personalizá-lo com foco em segurança. Por padrão, o Firefox coleta e armazena dados de uso e desempenho. Para cancelar essa prática, desative a coleta de dados de telemetria (isso será demonstrado). Passe algum tempo nas configurações de privacidade para ativar o bloqueio de pop-ups, a proteção anti-impressão digital (**anti-fingerprinting**) e a proteção contra phishing.

Eu particularmente utilizo mais de um navegador: o Firefox com várias configurações diferentes, o **Mullvad** para privacidade extrema e o **Brave** para um caso específico. Acredito que toda casa possui um idiota, e naturalmente ter um navegador comum me faz o idiota da casa nas horas vagas. O problema do Brave é seu núcleo Chromium, que, conforme já dito, é um prato cheio para hackers.

O principal subprojeto do **Tor Project** é, naturalmente, seu mecanismo de roteamento chamado Tor, mas esse projeto não é visível para o usuário comum. Na verdade, o contato do usuário comum com o Tor Project se faz por meio do **Tor Browser**, um segundo projeto que erroneamente é fixado como o carro-chefe do Tor Project. O Tor Browser consiste numa versão modificada do Mozilla Firefox Extended Support Release somada ao TorButton, ao TorLauncher, ao NoScript, à extensão HTTPS Everywhere e ao proxy Tor. Ele inicia automaticamente processos secundários do Tor e direciona o tráfego pela rede Onion. Ao fim de uma sessão, o navegador exclui todos os dados sensíveis de privacidade, como cookies HTTP e histórico de navegação — e é por isso que o usuário nem percebe que está usando o Tor. Cito o Tor Browser desde já porque ele é a **referência de hardening** deste mundo:

praticamente tudo que os outros navegadores "de privacidade" fazem é copiar, com menos rigor, o que o Tor Browser já faz por padrão. Sempre que houver dúvida sobre qual comportamento é o mais seguro, a pergunta útil é "o Tor Browser faria assim?".

Vou deixar a demonstração de uso de navegador para o capítulo sobre o Kodachi GNU/Linux, e minha recomendação final de uso para o tópico Navegadores em Identidade Hacker. Aqui o foco é entender o material de que são feitos e, principalmente, medir o que eles fazem sem a sua ordem.

Teste de privacidade baseado no privacytests.org

O objetivo do **PrivacyTests.org** é entender, em detalhes, quais dados cada navegador está vazando e quais navegadores oferecem as melhores proteções de privacidade. O PrivacyTests.org é uma iniciativa de código aberto que submete navegadores populares a um conjunto de testes automatizados. Esses testes são projetados para auditar as propriedades de privacidade dos navegadores da web de maneira imparcial. Os resultados são tornados públicos para ajudar os usuários a fazer uma escolha informada sobre qual navegador usar e para incentivar os fabricantes a corrigir vazamentos de dados privados.

Vale saber quem está por trás disso: o projeto foi criado e é mantido por **Arthur Edelstein**, pesquisador de privacidade que já trabalhou justamente no Tor Browser. O mérito do PrivacyTests é ser reprodutível — os testes são scripts abertos que qualquer um pode rodar — e neutro em relação a fabricante. Ainda assim, todo teste é uma foto de um momento: navegadores mudam de versão toda semana, e um resultado envelhece rápido. Os navegadores testados nesta rodada foram:

- Brave 1.67
- Chrome 126.0
- Firefox 127.0
- LibreWolf 127.0-1
- Mullvad 13.5
- Tor 13.5

O teste foi realizado em **2024-06-22 21:24:47 UTC** — reforço: é a foto daquele instante, com aquelas versões, e serve para entender **como** cada categoria de vazamento funciona, não para cravar rankings eternos. A seguir eu recolho os principais tópicos e sumarizo parte deles. Em cada tabela, as colunas seguem sempre a mesma ordem: **Brave, Firefox, LibreWolf, Mullvad, Chrome, Tor**.

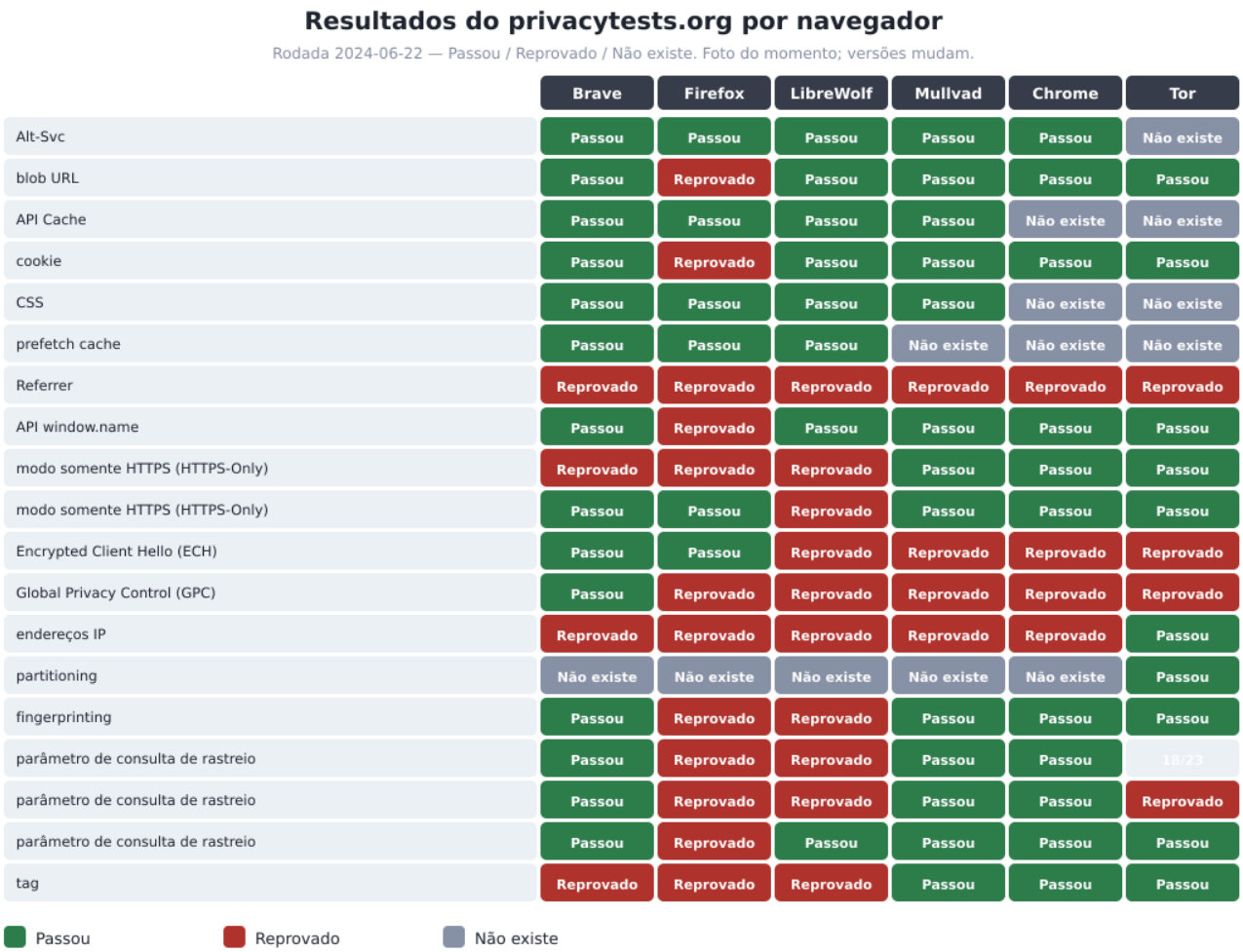

Figura: Resultados do privacytests.org por navegador (rodada 2024-06-22).

O **Alt-Svc** (Alternative Services) permite que o servidor indique ao navegador que um recurso deve ser carregado a partir de um servidor diferente. Por ser uma configuração persistente, pode ser usada para rastrear usuários entre sites se não estiver isolada corretamente.

O **blob URL** é uma referência local a alguns dados brutos. Os rastreadores podem usar um blob URL para compartilhar dados entre sites.

O **BroadcastChannel** foi projetado para enviar mensagens entre abas. Em alguns navegadores, pode ser usado para comunicação e rastreamento entre sites. Todos possuem o recurso e todos foram aprovados.

A **API Cache** é um mecanismo de armazenamento de conteúdo introduzido originalmente para dar suporte aos ServiceWorkers. Se o mesmo objeto Cache estiver acessível a vários sites, ele pode ser abusado para rastrear usuários.

Um **cookie** é uma pequena quantidade de dados armazenada pelo seu navegador em nome de um site. Ele tem usos legítimos, mas também é o mecanismo clássico de rastreamento entre sites e ainda hoje é o método mais popular de seguir usuários pela web. Os navegadores podem impedir que cookies sejam usados para rastreamento entre sites bloqueando-os e isolando-os (**partitioning**).

Os arquivos **CSS** são armazenados em cache e, se esse cache for compartilhado entre sites, pode ser usado para rastrear usuários. Todos possuem o recurso e todos foram aprovados.

O `navigator.storage.getDirectory` expõe um local para armazenar arquivos no conteúdo da web. Em alguns casos, esses arquivos podem ser compartilhados entre abas.

A **API IndexedDB** expõe um banco de dados transacional às páginas da web. Esse banco pode ser usado para rastrear usuários entre sites, a menos que seja isolado. Todos possuem o recurso e todos foram aprovados.

O recurso **prefetch cache** sugere ao navegador que ele busque um recurso com antecedência e o guarde em cache. Mas, se o navegador não isolar esse cache, ele pode ser usado para rastrear usuários entre sites.

O cabeçalho de requisição **Referrer** (`document.referrer`) é o mecanismo pelo qual o navegador informa a um site de onde o usuário está vindo. Esse cabeçalho rastreia inerentemente usuários entre sites. Recentemente os navegadores adotaram uma política de "cortar" o referenciador para transmitir menos informação de rastreamento, mas o Referer continua a transmitir dados de rastreamento entre sites por padrão. Todos foram reprovados.

A **API sessionStorage** é semelhante à localStorage, mas não persiste entre abas ou sessões. Ainda assim, pode ser usada para rastrear usuários que naveguem de um site a outro; esse rastreamento pode ser frustrado particionando o sessionStorage entre sites. Todos possuem o recurso e todos foram aprovados.

A **API window.name** permite que sites armazenem dados que persistem depois que o usuário navega a mesma aba para outro site. Esse mecanismo pode ser particionado para que os dados não persistam entre sites.

O próximo teste verifica se o navegador paralisa o carregamento de um site inseguro e avisa o usuário antes de dar a opção de continuar — o chamado **modo somente HTTPS (HTTPS-Only)** em alguns navegadores.

Outro par de testes verifica se um endereço inseguro digitado na barra é atualizado para HTTPS sempre que possível, e se um clique num hiperlink inseguro também é atualizado para HTTPS. Manter o usuário em HTTPS ajuda a mantê-lo seguro. Para ambos os itens o resultado foi:

O **Encrypted Client Hello (ECH)** é um protocolo novo que oculta, de bisbilhoteiros de rede de terceiros, qual site você está visitando — ele cifra a parte do handshake TLS que hoje ainda revela o nome do domínio.

O **Global Privacy Control (GPC)** é um cabeçalho HTTP que o navegador pode enviar para instruir um site a não vender os dados pessoais do usuário a terceiros. Um teste verifica se o cabeçalho GPC é enviado por padrão para o site de nível superior; outro verifica se o GPC é enviado para os elementos de terceiros na página.

Os **endereços IP** podem ser usados para identificar exclusivamente uma grande porcentagem de usuários. Um proxy, uma VPN ou o Tor podem mascarar o IP. A rede Tor envia as requisições do navegador por uma série de retransmissões para ocultar o IP, ajudando a mascarar identidade e localização; este teste verifica se a rede Tor está sendo usada por padrão. Dá para resolver isso com um `proxychains4`.

Os navegadores que usam Tor podem usar um circuito Tor diferente por site (**partitioning** do circuito). Mas o autor deste livro recomenda o WHONIX ou, se não houver recurso de memória, um `proxychains4`.

Agora entramos no terreno mais importante deste capítulo, o **fingerprinting** — a impressão digital do navegador. Imagine que seja possível obter uma telemetria do usuário sabendo as

dimensões da tela, o quanto o navegador está deslocado à direita e a distância do topo. Dá para inferir o sistema gráfico pelas barras. Páginas da web também podem detectar a presença de uma fonte instalada no sistema: a presença ou a ausência de várias fontes é comumente usada para gerar impressões digitais de usuários.

Vale abrir aqui, porque este é o conceito que menos gente entende e o mais poderoso. **Fingerprinting** é montar um identificador único a partir de pequenas diferenças de configuração que o seu navegador entrega sem pedir cookie nenhum. Ao contrário do cookie, que você pode apagar, a impressão digital renasce sozinha na próxima visita, porque ela é feita do que a sua máquina simplesmente é. Os principais vetores:

- **Canvas fingerprinting:** o site pede ao navegador para desenhar um texto ou uma forma num elemento `<canvas>` invisível e depois lê os pixels resultantes. Diferenças mínimas de GPU, driver, biblioteca de fontes e antialiasing fazem cada máquina render um resultado ligeiramente diferente — e esse resultado, resumido num hash, vira um identificador estável.
- **WebGL fingerprinting:** a mesma ideia, porém em 3D. O navegador expõe o modelo da placa de vídeo, o fabricante e o modo como ela renderiza uma cena — informação riquíssima para separar máquinas.
- **Fontes instaladas:** medindo a largura de textos ou consultando APIs, o site descobre quais fontes você tem. Sua lista de fontes é quase uma digital do seu perfil de software.
- **User-Agent e cabeçalhos:** a string de User-Agent, a lista de idiomas aceitos, os plugins, o fuso horário e a resolução — combinados — costumam bastar para individualizar você entre milhões.

A defesa correta não é "esconder" cada dado (isso muitas vezes só te torna mais raro, e portanto mais rastreável): é **parecer igual a todo mundo**. O projeto **Cover Your Tracks**, da **EFF (Electronic Frontier Foundation)** — sucessor do antigo Panopticlick —, mede exatamente isso: quantos bits de identificação seu navegador vaza e quão único você é na multidão. Vale rodar; muita gente descobre ali que é "único entre centenas de milhares", o que significa que nenhum cookie precisa existir para segui-la. É por essa mesma lógica que o Tor Browser tenta fazer todos os seus usuários parecerem a **mesma** máquina.

Ligada a essa ideia está a defesa das dimensões de tela. Para evitar o fingerprinting baseado no tamanho da janela, o navegador começa com uma janela de conteúdo arredondada, com espaços da ordem de 200px por 100px. A estratégia é jogar uma grande quantidade de usuários dentro de uma mesma resolução aparente. Os navegadores também trazem uma defesa de fingerprinting para os cenários de redimensionamento, chamada **Letterboxing** — uma técnica desenvolvida pela Mozilla e apresentada em 2019. Ela funciona adicionando margens (barras) à janela para que o conteúdo fique o mais próximo possível de um tamanho "padrão", comum a muitos usuários. Em palavras simples, o Letterboxing agrupa usuários em determinados tamanhos de tela, tornando mais difícil destacar alguém pelo tamanho da janela, já que muita gente aparentará ter a mesma tela.

Outro vetor é o **parâmetro de consulta de rastreio**. Quando você navega de uma página a outra, as empresas de rastreamento frequentemente enviam um parâmetro de consulta (query string) no endereço da segunda página. Esse parâmetro pode conter um identificador único que rastreia você individualmente enquanto navega, e costuma ser sincronizado com cookies, o que o torna um vetor poderoso. Os navegadores podem proteger você removendo parâmetros de rastreio conhecidos dos endereços antes de enviá-los. O conjunto de parâmetros testado pelo privacytests.org foi, em grande parte, obtido do próprio Brave, então

o autor acha o teste um tanto tendencioso a favor do Brave — mas reconhece os seguintes sinalizadores de URL: `_hsfp`, `hssc`, `hstc`, `s`, `hsenc`, `openstat`, `dclid`, `fbclid`, `gclid`, `hsCtaTracking`, `mceid`, `mkttok`, `mlsubscriber`, `mlsubscriberhash`, `msclkid`, `olyanonid`, `olyencid`, `rbclickid`, `scid`, `veroconv`, `veroid`, `wickedid` e `yclid`.

Quando uma página é visitada, ela frequentemente contém conteúdo de terceiros para rastreamento, como scripts e pixels de rastreamento. Esses componentes incorporados espionam você. Alguns navegadores e extensões mantêm uma lista de empresas de rastreamento e bloqueiam o carregamento desse conteúdo. Esta seção verifica se o navegador bloqueia 20 dos maiores rastreadores listados por `https://whotracks.me`.

Uma grande fração das páginas da web possui rastreadores de terceiros ocultos que leem e gravam cookies no seu navegador. Esses cookies podem ser usados para rastrear sua navegação entre sites. Esta seção verifica se o navegador interrompe o rastreamento entre sites por cookies de 20 dos maiores rastreadores listados por `https://whotracks.me`.

Uma vulnerabilidade comum é permitir que sites primários "marquem" (**tag**) seu navegador com alguns dados de rastreamento. Essa marca pode ser usada para reidentificar você quando retornar a um site já visitado. Essa categoria de vazamento pode ser evitada se o navegador limpar ou isolar os dados entre sessões.

O **Sistema de Nomes de Domínio (DNS)** é o método pelo qual os navegadores procuram o endereço IP de cada site que você visita. Numa consulta DNS, o navegador pergunta a um resolvedor (em algum lugar da internet) qual o IP correspondente a um nome de domínio (como `nytimes.com`). Tradicionalmente, a maioria dos navegadores envia essas consultas sem criptografia, o que significa que seu provedor (ISP) — ou qualquer um na rede entre seu computador e o resolvedor — pode espionar os sites que você visita. Nos últimos anos, navegadores e sistemas operacionais começaram a introduzir DNS criptografado, incluindo o **DNS sobre HTTPS (DoH)**, para cifrar a requisição e a resposta e evitar o vazamento do histórico de navegação. Estes testes verificam se um navegador ainda protege as consultas DNS, enviando-as criptografadas, mas sem alteração das configurações padrão. Você encontra neste livro uma descrição completa sobre segurança em DNS.

DNS criptografado por padrão — por grupo de resolvedor

	Brave	Firefox	LibreWolf	Mullvad	Chrome	Tor
Brasil, China, Alemanha e Índia	Reprovado	Reprovado	Reprovado	Reprovado	Passou	Passou
Rússia e EUA	Reprovado	Reprovado	Passou	Reprovado	Passou	Passou
Cloudflare e Google	Passou	Passou	Passou	Passou	Passou	Passou
Quad9	Reprovado	Reprovado	Passou	Passou	Passou	Passou

Figura: Proteção de DNS por padrão em cada navegador, por grupo de resolvedor.

Os pontos negativos acima podem ser trabalhados com adição de plugins e com mudanças no computador ou na infraestrutura. O autor deste livro não enxerga DNS criptografado (ou sobre TLS/HTTPS) como algo do navegador: **deve ser algo do sistema operacional** — porque, resolvido no sistema, vale para todos os processos, e não só para as abas do browser.

Lista de sites que rastreiam usuários na web

Um grupo que merece menção é o **Whotracksme**, que hoje também disponibiliza ferramentas de privacidade. Este tópico não é propaganda, e o autor deste livro não tem associação com nenhuma tecnologia ou empresa: o que será abordado é um filtro da produção desse grupo. Hoje o Whotracksme é conhecido como **Ghostery**, que tem como pilar a transparência de seus estudos. Para isso, coletam dados de sites e realizam pesquisas sobre navegadores com o objetivo de entender o quanto estamos anônimos quando navegamos.

A Ghostery recomenda que, para um navegador ficar mais coerente com a privacidade, ele tenha algumas extensões:

- **uBlock Origin**;
- **AdGuard**;
- **Adblock Plus**;
- **Privacy Badger**.

Todas serão mencionadas neste livro. Com esses componentes você reduz a chance de ser rastreado por sites como o `google.com`. As principais ferramentas de rastreio, na ordem de quebra de privacidade, são:

1. Google Tag Manager;
2. Google Static;
3. DoubleClick;
4. Google Fonts;
5. Google Analytics;
6. Google APIs;
7. Google;
8. YouTube (Audio/Video Player);
9. Google User Content;
10. Facebook;
11. Google Syndication/Advertising;
12. Amazon Advertising;
13. Amazon CloudFront;
14. Google Photos;
15. Cloudflare;
16. OneTrust (Consent Management).

Muitos serviços populares pertencem a Google, Amazon, Facebook (Meta), Apple e Microsoft, mas costumam ter nomes diferentes (por exemplo, a Meta possui Facebook, Instagram e WhatsApp). Os números são preocupantes: **74% dos dados de rastreamento de pessoas são controlados pelo Google**, e **33% dos mecanismos de rastreio envolvem cookies**. Há uma média de 12 requisições extras por página navegada só por APIs de rastreio embutidas em sites, e o número de ferramentas de rastreio por site chega a 7. Esses números expõem a grande preocupação dos hacktivistas que lutam por mais privacidade e anonimato.

Rastreadores por site (fonte: whotracks.me / Ghostery)

Site	Categoria	Trackers/página (média)	Total de rastreadores
google.com	Reference	3	13
youtube.com	Entertainment	7	14
reddit.com	Entertainment	3	15
facebook.com	Entertainment	2	10
amazon.com	E-Commerce	4	22
wikipedia.org	Reference	1	5
fiverr.com	Business	3	18
yahoo.co.jp	News and Portals	2	21
bing.com	Reference	2	18
twitter.com	Entertainment	3	10

Figura: Rastreadores por site segundo o whotracks.me / Ghostery.

A lista completa pode ser vista em `https://www.ghostery.com/whotracksme/websites`.

Telemetria de navegadores

Os dados da população são hoje o produto mais valioso para corporações e governos. Se um estudo minucioso for feito, pode-se mover multidões contra ou a favor de um tema ou alvo. Recentemente tivemos dois exemplos claros disso:

- **Internet e Revolução no Egito:** o uso de sites de redes sociais durante a convulsão social que derrubou o governo ditatorial egípcio em 2011;
- **Cambridge Analytica e Facebook:** o escândalo e suas consequências.

Se os sites atuam nesse filão de mercado, por que os navegadores não atuariam? Na onda contrária à privacidade e ao anonimato, temos navegadores que hoje coletam diretamente dados e os enviam como telemetria. Antes de continuar, convém definir o termo: **telemetria** é toda medição que um software faz de si mesmo e do seu uso e envia de volta ao fabricante — versão, falhas, tempo de uso, funcionalidades acessadas, às vezes muito mais. Há telemetria honesta (contar quantas vezes um botão travou) e telemetria abusiva (mapear seus hábitos). O problema é que, do lado de fora, você raramente distingue uma da outra: tudo que vê é tráfego saindo da sua máquina para um servidor da empresa, sem que você tenha clicado em nada.

Para este exemplo o autor deste livro desenvolveu um **proxy-fake** para capturar toda a telemetria de um navegador. Só de abrir o navegador, você verá a quantidade de requisições que são feitas sem a permissão do usuário. A lógica do experimento é simples e poderosa: um navegador recém-aberto, parado numa aba em branco, **não deveria falar com ninguém**. Se ele fala, alguém decidiu que ele fala por você.

Merece atenção especial um comportamento que vou destacar adiante no Firefox: a abertura de um **WebSocket** em segundo plano. HTTP comum é uma conversa de pergunta-e-resposta que termina; um **WebSocket** é um cano aberto e permanente, de mão dupla, entre o seu navegador e um servidor — pensado para chat e notificações em tempo real. Um navegador

ocioso que mantém um WebSocket vivo contra um servidor da empresa é, no mínimo, suspeito: é um canal por onde dados podem trafegar continuamente, nos dois sentidos, enquanto você acha que não está fazendo nada.

Em um Debian 12 com XFCE serão submetidos alguns navegadores sem auxílio de extensões ou bloqueadores — ou seja, uma instalação padrão. A ideia é desviar todo o tráfego para o proxy fake. Crie um arquivo chamado `proxy_fake.py`[^3] no diretório `/tmp` do Linux e edite o código abaixo.

[^3]: Código disponível no GitHub: `https://github.com/naoimportaweb/avulso/blob/main/proxy.py`.

```
  1 # script: /tmp/proxy_fake.py
105 import socket
106 import threading
108 index = 1
110 def handle_client_request(client_socket):
111     global index
112     request = b''
114     client_socket.setblocking(False)
115     while True:
116         try:
117             data = client_socket.recv(1024)
118             request = request + data
119         except:
120             break
121     host_string_start = request.find(b'Host: ') + len(b'Host: ')
122     host_string_end = request.find(b'\r\n', host_string_start)
123     host_string = request[host_string_start:host_string_end].decode('utf-8').strip()
124     if len(host_string.strip()) > 0:
125         print(str(index), "\t", host_string.strip())
126         index += 1
127     client_socket.close()
129 def start_proxy_server():
130     port = 8888
131     server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
132     server.bind(('127.0.0.1', port))
133     server.listen(10)
134     print(f"Proxy server listening on port {port}...")
135     while True:
136         client_socket, addr = server.accept()
137         client_handler = threading.Thread(target=handle_client_request, args=
(client_socket,))
138         client_handler.start()
140 if __name__ == "__main__":
141     start_proxy_server()
143 # export http_proxy='http://127.0.0.1:8888'
144 # export https_proxy='https://127.0.0.1:8888'
```

A ideia é alterar a configuração do navegador passando como proxy o IP local `127.0.0.1` e a porta `8888`, na qual estamos monitorando com o proxy fake. Ao fechar e abrir o navegador,

ele não deveria se conectar a nada; se conectar, então algo está sendo feito sem sua autorização ou ação. O truque do script é elegante justamente por ser burro: ele não completa a conexão para o destino real, apenas escuta o começo da requisição HTTP, extrai o cabeçalho `Host:` — que revela para qual domínio o navegador **queria** falar — imprime esse domínio numerado e fecha o socket. Nenhum tráfego chega ao destino; nós só espionamos a intenção. É um dedo-duro do navegador, contra o próprio navegador.

Não esqueça de iniciar o proxy fake. Para isso, execute em um terminal:

```
1 python3 /tmp/proxy_fake.py
```

Mozilla Firefox

O primeiro navegador não poderia ser outro. O Mozilla Firefox é tido hoje como o navegador que prega e executa a privacidade para seus usuários; legiões de hackers gritam "Firefox, Firefox!!!", então este será o primeiro.

Com o Firefox devidamente configurado com o proxy, feche e abra o navegador — sem editar nada, sem selecionar nada. Só observe ao fundo, no terminal, os domínios invocados pelo navegador sem a sua ação. Reconheço que fiquei triste, principalmente quando naveguei tentando entender cada uma dessas URLs. As principais são:

- `detectportal.firefox.com`
- `contile.services.mozilla.com`
- `push.services.mozilla.com`

Ao confirmar se a telemetria estava ligada, na URL `about:telemetry`, observei que estava tudo OK, que não havia telemetria. Mas, se algo sai por alguma requisição do navegador sem a ação humana, eu estou sendo monitorado. Veja que o Firefox estava com a telemetria desligada. Em versões pré-2024 a telemetria vinha ligada por padrão; já em 2024 ela vinha desligada por padrão — mas o navegador continua enviando requisições sem a ação humana.

Uma das URLs me chamou a atenção e, ao investigar, percebi que era um **WebSocket**. Aí vem a pergunta: qual o motivo que leva um navegador a abrir um WebSocket para uma empresa? Como expliquei acima, um WebSocket é um cano permanente de mão dupla — não é o tipo de coisa que um navegador ocioso deveria manter aberto contra o fabricante. Depois de tanto lutar, tenho que admitir: **o Mozilla Firefox não é mais um navegador que eu recomendo**, e vou gastar um tempo para migrar para outro.

Mullvad Browser

O Mullvad Browser foi um navegador que me recomendaram e de que realmente gostei; conforme já visto, foi bem avaliado. Mas vamos ver se ele é ruidoso quanto a requisições sem ação humana, e se é possível haver telemetria nessas comunicações analisando as URLs. Como de praxe, abra o Mullvad, configure o proxy, feche e abra novamente. Sem clicar em nada, observe por alguns minutos. As URLs invocadas sem a ação humana foram apenas duas:

1. `cdn.mullvad.net`;
2. `versioncheck-bg.addons.mozilla.org`.

A segunda existe por ele ser baseado em Gecko, e a primeira é um CDN com arquivos do navegador para atualização — pelo que entendi, ele se autoatualiza por esse CDN. Para desencargo de consciência, consulte `about:telemetry` para confirmar que a telemetria está desligada; veja que a propaganda da Mullvad é uma propaganda de privacidade. O Mullvad

pode ser um bom navegador: as únicas requisições feitas são de atualização, e ele leva em consideração os itens de privacidade do Tor Browser — claro, sem o uso da rede Onion.

LibreWolf Browser

Uma boa recomendação de um inscrito no canal foi o **LibreWolf**, um navegador que prega a privacidade e é baseado no Gecko. Porém, por usar a base do Mozilla, temos que esperar chamadas em segundo plano sem a ação humana. Em 10 minutos, as URLs invocadas foram:

- `https://statically.io/`
- `https://pgl.yoyo.org/`
- `https://curbengh.github.io/`
- `https://push.services.mozilla.com/`
- `https://about.gitlab.com/`
- `https://www.jsdelivr.com/`
- `https://curbengh.github.io/`

Ao pesquisar todos esses domínios, somente o `push.services.mozilla.com` é um risco — os demais são, em geral, fontes de listas de bloqueio e CDNs que o próprio LibreWolf usa para se manter atualizado.

Brave Browser

O **Brave** é um navegador aclamado, principalmente por quem usa Microsoft Windows ou por quem quer um Google Chrome "mais seguro". Para executar o Brave com proxy, abra antes o terminal, exporte as variáveis de ambiente `http*proxy*` *e* `*https*proxy` e, em seguida, chame o navegador:

```
1   export http_proxy='http://127.0.0.1:8888'
145 export https_proxy='https://127.0.0.1:8888'
147 brave-browser
```

Em pouco tempo começam as requisições sem a ação humana. As principais referências são:

- `variations.brave.com`
- `p3a-json.brave.com`
- `star-randsrv.bsg.brave.com`
- `safebrowsing.brave.com`
- `go-updater.brave.com`

Das URLs, a que é muito estranha é a `brave.com`: ao analisar os dados, trata-se de um **Base64** e, ao fazer o decode, encontra-se uma massa de dados criptografada. Aprenda: **tudo que é cifrado é sobre você e você não tem a chave — então é ruim para você.** Não recomendo o Brave, a não ser que você esteja no Microsoft Windows — pois aí você já seria um alvo para mim, kkkkk.

Google Chrome

De longe o **PIOR DOS MALWARES**. Vou até definir este navegador como um malware para a humanidade: extremamente vulnerável e uma máquina de telemetria contra o usuário. Para executar o Chrome com proxy, abra antes o terminal, exporte as variáveis de ambiente e chame o navegador:

```
1   export http_proxy='http://127.0.0.1:8888'
```

```
148 export https_proxy='https://127.0.0.1:8888'
150 google-chrome
```

Em pouco tempo começam as requisições sem a ação humana. As principais referências são:

- `clientservices.googleapis.com`
- `accounts.google.com`
- `update.googleapis.com`
- `optimizationguide-pa.googleapis.com`
- `beacons5.gvt3.com`
- `beacons.gvt2.com`
- `clients2.google.com`
- `safebrowsing.googleapis.com`

Esses arquivos `beacons` estão sempre trocando de URL — o que, por si só, já é um sinal de que não querem ser bloqueados nem catalogados.

Epiphany Browser

O **Epiphany** é um navegador de código aberto do projeto GNOME. No passado utilizava o Gecko como núcleo, mas, por dificuldades com a equipe do Mozilla, passou a usar o **WebKit**[^4] do projeto GTK. Para executá-lo com proxy:

[^4]: Acessível pela URL `https://webkitgtk.org/`.

```
  1 export http_proxy='http://127.0.0.1:8888'
151 export https_proxy='https://127.0.0.1:8888'
153 epiphany-browser
```

A pergunta é: cadê as requisições sem a ação humana? **ZERO ENVIO, TOTALMENTE SILENCIOSO — o único que PASSOU no teste.** Vale um adendo honesto: "não fazer telemetria" não significa "mais seguro em tudo". O Epiphany tem uma base de usuários pequena e um motor (WebKit) com menos investimento em blindagem de sandbox que o Chromium; ele ganha neste teste específico — o de não falar pelas suas costas —, e é exatamente esse o teste que este capítulo quis fazer.

Mecanismo de busca

Outro grande problema na vida de um hacker é pesquisar — visto que ninguém sabe tudo e, saiba, até os maiores hackers consultam algum lugar. O **Google Search** é uma máquina de anotar tudo que você pergunta, tudo que entra ou vê; em compensação, é o mecanismo que retorna os melhores resultados. Então o hacker deve se movimentar para outros serviços. O **DuckDuckGo** vem sendo bem avaliado, porém já noticiei no passado que ele também recolhe dados de usuários e está ligado a produtos de terceiros. Existem inúmeros outros que mudam conforme o tempo; peço que, quando estiver lendo este livro, procure os do seu momento.

Um mecanismo de busca pode ser executado de duas formas. A mais clássica hoje é o uso direto da pesquisa na barra de endereços. A segunda é abrir uma página de busca e digitar a pesquisa no corpo da página — e há diferença quando falamos de anonimato. Num navegador, algumas opções de busca são disponibilizadas, e ao comparar os navegadores você verá que terá outras opções. O engraçado é que todos operam da mesma forma por padrão: toda pesquisa vira parâmetro **query string** e então é conhecida tanto pelo seu navegador quanto pelas empresas de ISP. Ou seja, **não adianta ter um mecanismo de busca anônimo se você usa a barra de URL como campo de busca.**

Vamos usar como métrica alguns pontos para definir um bom mecanismo de busca:

- Não enviar query string;
- Não propor nada com IA;
- Não exibir propaganda (que é vetor para phishing);
- Ter as primeiras opções como URLs orgânicas;
- Número de elementos que podem ser bloqueados.

Vou iniciar pelo buscador do Google. Sem nenhum bloqueador de anúncios, veja que o Google faz tudo que é ruim para seus usuários. Quem acompanha as matérias hacker sabe: inúmeros ataques ocorrem com anúncios falsos em sistemas de busca, com sites clones dos oficiais que distribuem alguma aplicação trojanizada. O Google não avaliou nada — pagou, colocou, simples. Por isso o uBlock é tão importante para a segurança de todos: ele evita isso e lhe protege.

A resposta por IA automatizada é outro problema, visto que, em uma campanha em massa, ela pode "mudar de opinião", levando a massa para ideias manipuladas por governos. Estamos vendo o embate de Trump, que está querendo mandar na IA da Anthropic (fevereiro de 2025). Além da busca por query string na URL — que mantém o governo ciente da sua pesquisa —, o Google ainda adiciona hashes de marcação para lhe marcar e lhe seguir na internet; porém o governo também segue. No teste, os links bloqueados foram:

- `google.com`
- `accounts.google.com`
- `ogads-pa.clients6.google.com`
- `ogs.google.com`
- `play.google.com`
- `www.google.com`
- `csp.withgoogle.com`[^5]
- `googleadservices.com`[^6]
- `www.googleadservices.com`
- `googleusercontent.com`
- `lh3.googleusercontent.com`
- `gstatic.com`
- `encrypted-tbn1.gstatic.com`
- `encrypted-tbn2.gstatic.com`
- `fonts.gstatic.com`
- `ssl.gstatic.com`
- `www.gstatic.com`

[^5]: Análise completa: `https://pt.gridinsoft.com/online-virus-scanner/url/withgoogle-com`.

[^6]: Análise completa: `https://pt.gridinsoft.com/online-virus-scanner/url/googleadservices-com`.

O `googleadservices.com` é um domínio de propriedade do Google usado para exibir anúncios em vários sites e plataformas. Ele fornece uma plataforma para os anunciantes criarem e gerenciarem campanhas, segmentando públicos específicos e medindo a eficácia dos anúncios.

Em comparação, vou usar o **Disroot** como exemplo positivo de mecanismo de busca — o Disroot é famoso pelo seu ideal de anonimato. Se você o utilizar como mecanismo de busca em página, verá que ele não envia query string. Na verdade, o Disroot (bem como outros) faz as buscas por você nos mecanismos conhecidos e então traz os resultados. Imagine o Disroot

como um intermediário que o mantém anonimizado; na figura correspondente aparece até o nome do mecanismo que o Disroot utilizou. É possível ter o Disroot como mecanismo de busca do seu navegador, mas **não recomendo** — pois, usado assim, ele voltará a passar a consulta por query string, derrubando justamente a vantagem que tinha.

Extensões de navegadores

Para este tópico vou usar como parâmetro os navegadores usados no Kodachi GNU/Linux. Uma extensão de navegador é algo bom, pois facilita operações, mas, se mal escolhida, pode ser uma grande falha de segurança. Então escolha as extensões com sabedoria: mesmo que eu indique alguma, procure revisar o código-fonte do componente. Um simples clique no navegador e um resumo de localização e endereços já podem ser observados — e isso é importante, na verdade fundamental que se faça constantemente. Esse procedimento com uma extensão é muito mais fácil do que navegar até um site. O componente que uso para isso é o **IPWHOIS.io**, muito utilizado, embora pouco avaliado.

Antes de indicar extensões, vale mostrar **como auditar uma extensão que já está instalada**, porque toda extensão roda com privilégios altos dentro do navegador — ela pode ler o que você digita. Para obter informações sobre as extensões e os addons, no diretório do navegador dentro da área do usuário, procure a pasta do Firefox. Numa distribuição Kodachi GNU/Linux, por exemplo, o diretório é `~/.mozilla/firefox/5vr1hbth.KodachiBrowser`. Os dados dos addons estão no arquivo `addons.json`; pegue esse arquivo e abra-o em algum visualizador de JSON, afinal JSON não é um arquivo feito para leitura humana. Ali estão o nome e os detalhes da extensão. Pegue o **ID** — o ID é o nome do arquivo que vamos obter para validar.

O arquivo estará em `~/.mozilla/firefox/5vr1hbth.KodachiBrowser/extensions/{3cc4ec19-12d7-4081-9d88-a6082b71e5ec}.xpi`. Um `.xpi` é apenas um pacote (na prática, um ZIP) com o código da extensão. Pegue esse arquivo e submeta-o ao **virustotal.com** — no exemplo, não foi localizado nenhum problema. O VirusTotal passa o arquivo por dezenas de motores antivírus de uma vez; ele não é infalível, mas é uma primeira peneira barata antes de confiar código de terceiros dentro do seu navegador. Recomendo sempre analisar as extensões oficiais; para isso acesse `https://addons.mozilla.org/en-US/firefox/extensions/category/privacy-security/`. Vamos analisar algumas.

A extensão **HackBar** possui um conjunto de funcionalidades que vão desde validações de segurança até ataques propriamente ditos, e ainda é gratuita. Recursos do HackBar:

- É útil para verificar a segurança de aplicativos e servidores web;
- É usada por pesquisadores de segurança;
- Pode ser usada para verificar vulnerabilidades de Cross-Site Scripting (XSS) no site;
- Pode ser usada para verificar vulnerabilidades de SQL Injection no site;
- Pode ser usada para encontrar subdomínios de sites.

A extensão **uBlock Origin** é uma extensão multiplataforma, livre e de código aberto, usada para filtragem de conteúdo, incluindo bloqueio de anúncios. Está disponível para diversos navegadores:

- Chrome;
- Chromium;
- Edge;
- Firefox;
- Opera;

- Safari.

O uBlock Origin recebeu elogios de sites de tecnologia e especialistas em segurança, pois consome pouca memória se comparado com outras extensões. O objetivo declarado do uBlock Origin é fornecer aos usuários os meios de impor suas próprias escolhas de filtragem de conteúdo. Rapidamente ganhando força como ferramenta de bloqueio de anúncios, o uBlock no Firefox chegou a 5 milhões de usuários ativos e, no Chrome, posteriormente ultrapassou 10 milhões. O desenvolvimento (com o desenvolvedor Nik Rolls) chegou ao Microsoft Edge em dezembro de 2016. Em janeiro de 2017, o uBlock Origin foi adicionado aos repositórios do Debian 9 e do Ubuntu (16.04), e a extensão foi premiada por editores como uma das melhores do mês pela Mozilla.

Para exemplificar, basta abrir uma página web e clicar no símbolo do uBlock na barra superior: uma nova janela popup abre e, em tempo de execução, baseado em uma lista de URLs perigosas, ele vai bloqueando requisições. No caso de uma página comum, nenhum link é danoso. Ao clicar em "Mais", a ferramenta exibe mais dados; no caso do Google Docs, há mais elementos bloqueados — lógico, o Google Docs é da Google, a empresa que mais coleta dados de usuários. Onde:

1. Quais são as URLs que foram bloqueadas;
2. Mais informações visíveis;
3. Menos informações visíveis.

A extensão **Penetration Testing Kit (PTK)** é uma solução completa para simplificar tarefas diárias no campo da segurança de aplicativos. Ferramenta fundamental para profissionais de segurança, ela foi projetada para melhorar a eficiência e fornecer informações valiosas para o cotidiano. Basta abrir o navegador, ir até a URL que quer testar e executar a extensão; a ferramenta demora alguns segundos e começa a mapear dados.

A verificação em tempo de execução no navegador oferece Testes Dinâmicos de Segurança de Aplicativos (**DAST**) e Análise de Composição de Software (**SCA**) diretamente no navegador. Detecta Injeções SQL, Injeções de Comando, vulnerabilidades de Cross-Site Scripting (XSS) armazenado e refletido, e muito mais. Identifica ainda ameaças complexas como bypass de autenticação SQL, injeções XPath e ataques a JWT. O **JWT Inspector** adicionou uma funcionalidade crucial que permite analisar JSON Web Tokens (JWT), criar novos tokens e gerar chaves públicas e privadas para assinatura JWT.

Você obtém acesso, com um clique, a informações detalhadas sobre o aplicativo alvo, incluindo pilha de tecnologia, Firewalls de Aplicativos Web (WAFs), cabeçalhos de segurança, links rastreados e fluxo de autenticação. O PTK inclui um proxy com log de tráfego detalhado; esse log permite repetir qualquer requisição no **R-Builder** ou enviá-la ao **R-Attacker**. Você pode automatizar a execução de XSS, injeção SQL ou injeção de comandos de sistema. A extensão inclui o R-Builder, ferramenta poderosa para criar e manipular requisições HTTP com precisão — inclusive ataques de contrabando de requisições (**HTTP request smuggling**) —, além de um editor de cookies (adicionar, editar, remover, bloquear, proteger, exportar e importar) e um utilitário de codificação/decodificação para vários formatos, incluindo UTF-8, Base64 e MD5. Esta extensão será usada na aula de Vulnerabilidades Web.

O **Wappalyzer** é uma extensão multiplataforma que, ao ser usada, revela as tecnologias utilizadas pelos desenvolvedores no site alvo. Ele detecta sistemas de gerenciamento de conteúdo, plataformas de comércio eletrônico, frameworks web, software de servidor, ferramentas analíticas e muito mais. Assim como o PTK, essa extensão exibe informações sobre o site, principalmente tecnologias usadas — o que pode ser importante, pois direciona

uma estratégia hacker contra um alvo. Geralmente um hacker acessa um site alvo, navega e analisa antes de montar a estratégia. Por exemplo, ao acessar um site que usa WordPress com vários componentes, percebe-se que é mais um daqueles sites configuráveis que saem de fábrica cheios de vulnerabilidades. Uma funcionalidade interessante é a exportação: a ferramenta exporta um CSV que podemos importar em outras ferramentas.

O **Cookie Quick Manager** é uma extensão que se propõe a ser um poderoso gerenciador de cookies, principalmente daqueles acumulados durante a navegação. A extensão permite visualizar, editar, criar, excluir, fazer backup, restaurar cookies e pesquisar por nomes de domínio. Além disso, o LocalStorage da página visualizada pode ser excluído. Foi projetado para desenvolvedores, testadores ou pessoas preocupadas com sua privacidade na internet — principalmente nós, hackers, que somos muito preocupados com a internet. Funcionalidades básicas:

- **User friendly:** interface clara e estruturada. Cada parâmetro e funcionalidade é descrito quando o mouse está sobre o elemento; escolha a abertura em uma aba para uma visão mais ampla;
- **Search:** o usuário pode pesquisar cookies de um domínio e dos subdomínios que dependem dele;
- **Edit/Create:** todos os atributos de um cookie podem ser modificados — domínio, caminho, nome, valor, data de expiração, além dos sinalizadores `secure` e `httponly`;
- **Delete:** remove os cookies do site atual em dois cliques;
- **Export:** exportação e importação de um ou vários cookies, de um ou vários domínios, no formato JSON ou Netscape;
- **First-Party Isolation:** suportado com algumas limitações (por bugs de API) no Firefox 59, 60 e 61, e sem limitações no Firefox 62;
- **Contexts:** contextos (também chamados de Multi-Account Containers ou Contextual Identities) são suportados — dá para pesquisar e exibir cookies dentro de um contêiner, copiar cookies de um contêiner para outro ou salvar um cookie em um contexto específico;
- **SameSite:** proteção parcial contra os riscos de Cross-Site Request Forgery (CSRF) e Cross-Site Script Inclusion (XSSI), implementada desde o Firefox 63;
- **Cookie protection:** exclui cookies, exceto os protegidos, com dois cliques, a qualquer momento, do site que você está visualizando; uma opção também pode impedir que os cookies sejam excluídos pelos próprios sites;
- **Protection of session cookies:** os cookies de sessão podem ser protegidos em dois cliques para evitar logout acidental após a limpeza de cookies normais;
- **Cleaning and privacy:** pode excluir automaticamente todos os cookies na inicialização;
- **LocalStorage:** chaves/valores da página visualizada podem ser excluídos.

O código-fonte é gratuito (sob **GPLv3**) e publicado em uma plataforma pública — a única maneira de permitir revisões e contribuições externas. Ao abrir um site, você pode operar sobre o site atual ou sobre todos os cookies. Selecione **Manage all Cookies** e abrirá uma janela com muita informação; escolha um site que visitou recentemente e verá os cookies exibidos à direita. Como já dito, você pode editar o cookie, salvar uma exportação em arquivo ou até importar um cookie salvo. Um hacker precisa desses cookies salvos para realizar ataques a interfaces web que usam cookies para guardar dados de sessão: com esses cookies em mãos, dá para automatizar os ataques com a biblioteca `requests` do Python. Onde:

1. Filtro e busca de site/domínio;

2. Dados do cookie, que podem ser editados;
3. Exportação, importação, bloqueio de edição, etc.

O **NoScript Security Suite** é uma extensão compatível com a grande maioria dos navegadores modernos. É um software de distribuição livre, criado originalmente por **Giorgio Maone**, que a comunidade rapidamente abraçou pela necessidade de uma ferramenta capaz de parar a execução de scripts indesejados feitos em JavaScript, Java Applets, Flash, Silverlight e outros. Além dessa base de conhecimento sobre scripts maliciosos, o usuário pode adicionar URLs a listas. A ferramenta evoluiu de tal forma que é usada hoje até como contramedida contra exploits de navegadores — não à toa, é uma das peças que compõem o Tor Browser. Atenção: o site oficial é `https://noscript.net`.

Ao entrar em um site, você deve analisar os scripts presentes. Por exemplo, o domínio `cryptobrowser.site` (que não é do autor deste livro) pode ser um site ruim, com mineradores de criptomoeda embutidos no navegador — então analise os scripts. Clique no símbolo de negar e verá que aquele script deixa de ser confiável: sempre que for carregado, o sistema irá excluí-lo. Uma boa pedida para privacidade é ter, rodando em conjunto, o uBlock e o NoScript — ambos formam uma boa dupla de extensões.

Ferramentas citadas

- **uBlock Origin** — bloqueador de conteúdo e anúncios eficiente e leve, multiplataforma; **código aberto** (GPLv3).
- **NoScript Security Suite** — bloqueia por origem a execução de scripts (JavaScript, Flash, etc.), criado por Giorgio Maone; **código aberto** (GPL).
- **Mullvad Browser** — navegador de privacidade baseado em Firefox/Gecko, feito com o Tor Project sem usar a rede Onion; **código aberto** (MPL).
- **LibreWolf** — fork do Firefox com telemetria removida e privacidade reforçada; **código aberto** (MPL).
- **Brave** — navegador baseado em Chromium com bloqueio embutido; **código aberto** (MPL 2.0), com componentes de anúncios/telemetria próprios.
- **Epiphany (GNOME Web)** — navegador do projeto GNOME sobre WebKitGTK; **código aberto** (GPL).
- **Tor Browser** — Firefox ESR modificado com Tor e NoScript, referência de hardening; **código aberto**.
- **Disroot** — coletivo de serviços livres (inclui metabusca intermediada); **código aberto**, mantido por associação sem fins lucrativos.
- **Wappalyzer** — identifica as tecnologias de um site (CMS, frameworks, servidores); a versão de detecção local foi **código aberto (MIT)** no passado, hoje com produto **comercial**.
- **HackBar** — barra de testes de segurança web (XSS, SQLi) para o navegador; **gratuita** (freeware; versões pagas/comerciais).
- **Penetration Testing Kit (PTK)** — suíte de testes DAST/SCA no navegador, com proxy, R-Builder e R-Attacker; **gratuita** (freeware comercial).
- **Cookie Quick Manager** — gerenciador completo de cookies e LocalStorage; **código aberto** (GPLv3).
- **VirusTotal** — serviço online que passa arquivos/URLs por dezenas de antivírus; **serviço comercial** (Google/Chronicle), com camada gratuita.
- **IPWHOIS.io** — extensão/serviço de consulta rápida de IP e localização; **serviço comercial** com camada gratuita.

Referências

- EDELSTEIN, Arthur. **PrivacyTests.org: open-source tests of web browser privacy**. Teste de 2024-06-22. Disponível em: `https://privacytests.org/`. Acesso em: jul. 2025.
- GHOSTERY / WHOTRACKS.ME. **WhoTracks.Me: transparency of web tracking**. Disponível em: `https://www.ghostery.com/whotracksme/` e `https://whotracks.me/`. Acesso em: jul. 2025.
- ELECTRONIC FRONTIER FOUNDATION (EFF). **Cover Your Tracks** (sucessor do Panopticlick): browser fingerprinting e proteção. Disponível em: `https://coveryourtracks.eff.org/`. Acesso em: jul. 2025.
- MOZILLA. **Firefox fingerprinting protection and Letterboxing** (2019). Disponível em: `https://support.mozilla.org/`. Acesso em: jul. 2025.
- TOR PROJECT. **The Design and Implementation of the Tor Browser**. Disponível em: `https://2019.www.torproject.org/projects/torbrowser/design/`. Acesso em: jul. 2025.
- MOZILLA. **Telemetry — Firefox Source Docs** (`about:telemetry`). Disponível em: `https://firefox-source-docs.mozilla.org/`. Acesso em: jul. 2025.

Sistema de Arquivos Cifrado

É fundamental que o seu sistema de arquivos esteja protegido. Você trabalhará com arquivos e artefatos que, em caso de uma prisão, poderão ser usados contra você — então é fundamental saber criptografar e eliminar vestígios do seu computador. A princípio você deve criptografar o disco, conforme o tópico "Protegendo a Máquina Virtual", mas precisa usar mais uma abordagem sobre esta. Existem duas boas alternativas: uma é o uso do **VeraCrypt** e outra é o **CodóEncrypt**. Mas sempre use um Tor, I2P ou VPN associado.

A prisão é sempre uma sombra que acompanha o hacker, e você deve estar preparado. Vão realizar forense nos seus equipamentos, e para isso vão usar softwares aos quais você jamais teria acesso: o governo é forte, o governo tem dinheiro e o governo quer acabar com a sua vida (lembre-se do Leviatã). Mas mesmo que esteja no vale à sombra da morte, seja fiel à sua criptografia, pois ela lhe salvará. Você deverá criar a sua **defesa em profundidade (defense in depth)**, em níveis, conforme o esquema a seguir.

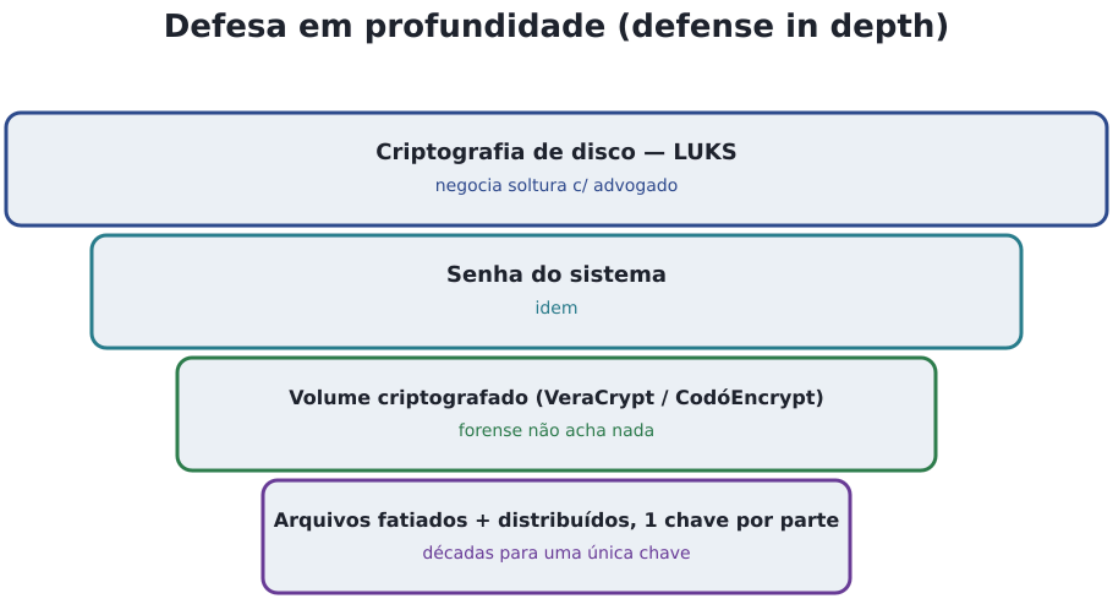

Figura: As camadas da defesa em profundidade.

A primeira criptografia é a **criptografia de disco (full-disk encryption)**: o volume do seu disco deve estar criptografado, e para isso usamos o **LUKS**. Em seguida vem a senha do seu computador. Essas duas barreiras serão usadas por você para negociar a sua soltura imediata: você as entregará mediante a postura de "estou concordando e quero ajudar na investigação". Seu advogado saberá conduzir isso, então deixe para ele. Quando a forense for iniciada dias depois, nada será achado — a não ser arquivos enormes e criptografados. Dentro deles estarão os arquivos sigilosos do hacker. A partir daí você não será mais responsável por ajudar, afinal já colaborou com as investigações, e o seu advogado irá lhe assistir neste segundo momento.

Como o LUKS funciona por dentro

Antes de operar as ferramentas, vale entender o alicerce sobre o qual toda essa defesa se apoia. O **LUKS (Linux Unified Key Setup)** não é um algoritmo de cifra, e sim um **formato**

padronizado de cabeçalho que fica no início do dispositivo criptografado e organiza como a chave é guardada. Por baixo dele opera o **dm-crypt**, o subsistema do kernel Linux que cifra e decifra os blocos do disco em tempo real, de forma transparente — você monta o volume, digita a senha e trabalha normalmente, enquanto cada setor é decifrado só quando lido e recifrado quando escrito.

O ponto engenhoso do LUKS está na separação entre a sua senha e a chave que realmente cifra o disco. A chave que cifra os dados é a **master key**, um valor aleatório longo que você nunca vê nem digita. O cabeçalho LUKS guarda essa master key **cifrada** em até oito **keyslots**, e cada keyslot pode ser aberto por uma senha (ou arquivo-chave) diferente. Ao digitar a senha, o LUKS a passa por uma função de derivação lenta — **PBKDF2** no LUKS1, **Argon2** no LUKS2 — que consome tempo e memória de propósito, encarecendo o ataque de força bruta. Só depois de derivada é que a chave abre o keyslot, revela a master key e libera o volume. Isso tem duas consequências práticas que você precisa gravar: primeiro, trocar a senha é instantâneo, porque só reembala a master key, sem recifrar o disco inteiro; segundo, se o **cabeçalho for corrompido ou apagado**, a master key se perde para sempre e nem você recupera os dados — por isso convém fazer um backup do cabeçalho em local seguro. Os comandos básicos giram em torno do utilitário `cryptsetup`:

```
# criar (formatar) um volume LUKS em um dispositivo ou arquivo
sudo cryptsetup luksFormat /dev/sdX

# abrir o volume (mapeia para /dev/mapper/segredo)
sudo cryptsetup luksOpen /dev/sdX segredo

# fechar quando terminar
sudo cryptsetup luksClose segredo

# fazer backup do cabeçalho (guarde longe do disco!)
sudo cryptsetup luksHeaderBackup /dev/sdX --header-backup-file header.img
```

Entender isso importa porque as duas ferramentas a seguir se apoiam nesse mesmo mecanismo. E há um ponto que o autor faz muito bem em destacar na estratégia jurídica: a criptografia de disco é a camada que você **pode** entregar ao negociar. O que interessa manter escondido são as camadas de dentro.

VeraCrypt

O **VeraCrypt** é o software de criptografia mais conhecido, disponível tanto para a plataforma Microsoft quanto para Linux, capaz de criptografar diretórios e pendrives. A parte boa é que possui uma interface amigável — ótimo para usuários leigos que não querem entrar em detalhes técnicos. É uma ferramenta gratuita e de código aberto. O VeraCrypt é um **fork** de uma solução já descontinuada, o **TrueCrypt** (e o VeraCrypt mantém compatibilidade com volumes TrueCrypt), com mais de doze anos de uso — o projeto começou em 2013. O ideal é utilizar o `apt` para fazer a instalação, com os dois comandos abaixo:

```
sudo apt update -y
sudo apt install veracrypt -y
```

Mas nem sempre você terá o VeraCrypt no repositório principal. Vejo inúmeros hackers criando **ClickFix** para enganar aspirantes, sempre ensinando a instalar a partir de repositórios falsos, adicionando fontes falsas ao `sources.list`. Então, se não tiver o pacote, você deverá instalar por **DPKG** ou por **AppImage**. Neste livro vou usar o AppImage, pois no meu dia a dia uso o CodóEncrypt, por motivos que explicarei adiante. Mas o VeraCrypt é um

aplicativo que você tem que conhecer e usar, se não puder usar outra coisa. Talvez precise, antes, instalar o **libfuse2**, com o comando abaixo:

```
sudo apt install libfuse2
```

Execute o VeraCrypt e aparecerá uma interface para adicionar volumes. Provavelmente você ainda não tem nenhum volume, então vamos criar um.

O fluxo do assistente é direto. Clique em **Create Volume**. Selecione o básico mesmo — um arquivo criptografado (**encrypted file container**). Crie um diretório e dê o nome do arquivo; aqui estou dando muita bandeira ao usar uma extensão `.img` (um nome discreto denuncia menos). O primeiro passo técnico é escolher a criptografia: **AES** é uma boa cifra, muito forte, e não será quebrada por computadores comuns — vale lembrar que o VeraCrypt não traz algoritmos pós-quânticos. Informe um tamanho para o volume, e o VeraCrypt criará uma espécie de diretório com esse tamanho. Informe uma chave muito forte, bem complicada, mas que você não esqueça — **não anote**, cuidado total aqui. Use sempre **FAT** como sistema de arquivos do volume; a segurança será dada pelo VeraCrypt. Por fim, será preciso ficar passando o mouse aleatoriamente pela tela: o objetivo é gerar **entropia (randomicidade)** para a geração das chaves.

Negação plausível e volumes ocultos

Há um recurso do VeraCrypt que o manuscrito não detalha, mas que é o coração do que essa ferramenta oferece de mais poderoso, e que se encaixa perfeitamente na estratégia jurídica descrita no início: a **negação plausível (plausible deniability)** por meio de **volumes ocultos (hidden volumes)**.

A ideia é a seguinte. Dentro do espaço "vazio" de um volume VeraCrypt comum (o volume externo), você pode criar um segundo volume, escondido, com **outra senha**. Como todo o espaço livre de um volume VeraCrypt é preenchido com dados aleatórios desde a criação, é **matematicamente impossível** provar, olhando o container, que existe um segundo volume ali dentro — dados cifrados e ruído aleatório são indistinguíveis. Na prática, você passa a ter duas senhas para o mesmo arquivo: uma senha revela o volume externo, com arquivos plausíveis mas não comprometedores (fotos, documentos banais); a outra senha revela o volume oculto, onde estão os arquivos reais. Sob coação — o chamado ataque de **rubber-hose** (a "mangueira de borracha", ou seja, tortura e coerção) — você entrega a senha do volume externo, e não há como o adversário demonstrar que existe mais alguma coisa.

É a versão criptográfica exata da tática que o autor descreve: entregue o que é negociável, mantenha o que importa invisível. Uma ressalva honesta: a negação plausível é forte contra a análise do disco, mas frágil contra vazamentos do sistema operacional (arquivos recentes, miniaturas, logs que registrem o volume oculto montado). O VeraCrypt oferece um modo de sistema operacional oculto justamente para mitigar isso, mas a disciplina de opsec continua sendo sua.

CodóEncrypt

Neste livro será demonstrado o uso do **CodóEncrypt** e explicada a arquitetura do sistema. A demonstração será baseada na versão 1.001, disponível para download gratuitamente. Nela, alguns problemas de anti-forense já foram resolvidos, tais como:

- Armazena arquivos com **nomes aleatórios**;
- Cria um **diretório virtual criptografado com LUKS**, e o usuário não sabe a chave;
- Mantém o **journaling** dentro do diretório criptografado;

- Criptografa **cada arquivo com uma chave criptográfica diferente**;
- Corta os arquivos em pedaços e envia para **diferentes repositórios**;
- Utiliza criptografia **"AES 512"**;
- Pode ser utilizado para troca de arquivos entre máquinas diferentes.

Entenda a arquitetura do sistema. Para exemplificar, imagine que você vai escrever um dossiê utilizando o LibreOffice. Para isso, deve estar em execução na máquina, como serviço, uma aplicação chamada **Local Service** — é o componente do CodóEncrypt responsável por abrir e gerenciar um **disco virtual criptografado**, e é dentro desse disco que ficará o dossiê, ou seja, o arquivo com extensão `.odt`.

Para não exigir muito esforço, foi desenvolvida uma interface gráfica, a **Codó GUI**, que envia comandos ao Local Service, o qual opera sobre os arquivos criptografados. O Local Service armazena pedaços do arquivo (o dossiê) devidamente criptografados em serviços remotos, que podem estar dentro da própria rede ou fora da rede local. Lembre-se de que cada arquivo é criptografado com uma chave diferente, cada arquivo é cortado em pedaços, e cada pedaço recebe um **nome aleatório de 64 caracteres**. Mesmo que a máfia, por intermédio da ABIN, obtenha as partes guardadas remotamente, não terá capacidade de descriptografar. E, se você cadastrar mais de um serviço, os pedaços serão distribuídos entre eles, dificultando ainda mais o trabalho — por isso recomenda-se cadastrar mais de um serviço remoto. Para reconstruir os dados, a forense teria de:

1. Acessar todos os seus serviços remotos;
2. Obter todos os pedaços de arquivos;
3. Organizar todos os pedaços — mas eles estão com nomes aleatórios;
4. Para cada arquivo, se os pedaços tiverem sido organizados corretamente, iniciar então um longo processo de décadas para achar a chave do arquivo — e isso apenas se acertarem a sequência dos pedaços.

Pela interface Codó GUI, você inicia o trabalho carregando um **arquivo de definição** que contém o endereço de cada arquivo e de cada pedaço. É claro que esse arquivo, que fica na sua máquina, está criptografado com "AES 512" — e esse é o único ponto falho. Você deve esconder esse arquivo: pode ser num pendrive criptografado com VeraCrypt ou num Kingston AES. Na versão 2, esse arquivo será escondido por meio de técnicas de **Polyfile**. Sempre que decidir parar de trabalhar, salve e feche o projeto no Codó GUI.

Uma nota honesta sobre "AES 512"

Aqui é preciso um esclarecimento técnico, feito com respeito ao trabalho do autor, porque a expressão "AES 512" aparece várias vezes e um leitor atento vai estranhar. O **AES (Advanced Encryption Standard)**, padronizado pelo NIST em 2001, **não existe** com chave de 512 bits. O padrão define exatamente três tamanhos de chave: **128, 192 e 256 bits**. Não há um "AES 512" oficial — quem procurar por esse termo na especificação não o encontrará.

Isso não significa que a ferramenta seja fraca ou que o autor esteja enganando alguém. Há explicações técnicas legítimas para o rótulo, e a mais provável é uma destas: (a) uma **cascata de cifras (cipher cascade)**, em que os dados passam por AES-256 mais de uma vez com chaves independentes — algo que o próprio VeraCrypt oferece em modos como AES-Twofish-Serpent; (b) uma **chave-mestra derivada de 512 bits** que é então dividida ou usada em dois blocos de 256 bits (o modo **XTS**, usado em cifragem de disco, de fato consome uma chave do dobro do tamanho — o "AES-256-XTS" usa 512 bits de material de chave); ou (c) simplesmente uma nomenclatura própria do projeto para indicar "o mais forte possível". Qualquer uma dessas leituras é defensável. O que **não** se sustenta é imaginar que exista uma

variante do algoritmo AES operando internamente com blocos ou chave de 512 bits fora do que o NIST padronizou. Fica o registro: a segurança real vem de AES-256 bem implementado (já mais do que suficiente contra qualquer adversário clássico), e "512" é, quase certamente, uma referência a encadeamento ou a material de chave estendido — não a um algoritmo novo.

Instalando o CodóEncrypt

Comece baixando, em um **Debian GNU/Linux**, o arquivo de instalação. O link da página é `https://sourceforge.net/projects/codoencrypt/files/`. Faça o download no seu Debian e salve o arquivo em `~/Downloads`; depois extraia para o mesmo diretório. Abra um terminal em `~/Downloads` e navegue para dentro do diretório `codoenc-main/local`. Em seguida, execute o processo de instalação com `sudo` — é simples, basta rodar o script `install.sh`:

```
  1 cd ~/Downloads/codoenc-main/local
165 sudo ./install.sh
```

Aguarde até o fim da instalação. Nesse trecho final será criado o serviço **Local Service**, conforme já descrito, e ele já será configurado para iniciar automaticamente junto com o seu Linux. Agora é a hora de instalar o segundo programa, a **Codó GUI**. Para isso, navegue para `~/Downloads/codoenc-main/app` e execute o segundo `install.sh`:

```
  1 cd ~/Downloads/codoenc-main/app
166 sudo ./install.sh
```

Esse processo será mais complexo, pois deverá instalar o pacote gráfico Python — o que demora, já que fará um download de aproximadamente 250 MB. Agora verifique se o serviço foi iniciado, executando o comando abaixo e observando o output: veja que está ativo e rodando.

```
1 sudo systemctl status kfm_codo.service
```

O disco virtual criado tem aproximadamente 1,9 GB. Esse disco virtual — já explicado — fica em `/tmp`, e você **não pode saber a chave de descriptografia** dele; por isso ela não lhe foi solicitada. Você pode configurar apenas o **tamanho** do disco virtual. Para isso, abra para edição, com o `nano`, o arquivo de configuração:

```
1 sudo nano /opt/codoencrypt/local/data/config.json
```

Em `size`, você pode aumentar ou diminuir o tamanho desse diretório virtual — basicamente, é 1 para cada 120 MB aproximadamente. Lembre-se de que um valor muito grande consumirá muito disco e demorará mais, pois ele criptografa todo o espaço. Pagamos um preço pela nossa segurança. Recomendo, para reduzir o tempo, o uso de SSD ou SSD M.2; disco rígido é impraticável. Sempre que modificar o arquivo acima, será preciso reiniciar o serviço:

```
1 sudo systemctl restart kfm_codo.service
```

Criando o primeiro projeto

Agora vamos executar a parte gráfica. Em qualquer ponto do sistema de arquivos, invoque o comando `codog`. Uma janela será aberta sem nenhum projeto; como estamos instalando agora, vamos criar um novo projeto em **Novo**. Para criar um novo projeto, temos que informar um diretório que será usado para armazenar o arquivo de definição criptografado e também um nome para o projeto. Uma **chave** deve ser informada — será usada para criptografar com "AES 512". Recomendo que consiga criar e decorar uma chave de 32 caracteres; se informar

uma chave menor, o sistema irá concatená-la consigo mesma até alcançar os 32 caracteres obrigatórios. Esse arquivo será o seu ponto fraco, então seja muito criterioso com essa chave.

Por padrão, o serviço rodará na **porta 50001**, mas isso pode ser trocado no arquivo `/opt/codoencrypt/local/data/config.json`, conforme já demonstrado. A interface é dividida em duas áreas: uma **Treeview**, na qual você verá os seus arquivos, e um painel de detalhes com as opções sobre cada arquivo.

O primeiro passo é definir quais são os servidores. Para isso, vá em **configurações do projeto** e, na aba de **Servidores**, clique no botão **Adicionar**. Para este exemplo, previamente foram enviados os arquivos para um serviço Apache2 na máquina `192.168.1.135`, que tenho na minha rede local:

- `upload.php`
- `download.php`
- `status.php`

Você pode alugar um serviço HTTP na Internet e adicionar esses arquivos — nos tópicos seguintes descreverei esse processo. Recomendo que tenha muitos servidores em locais diferentes, e ainda que diversifique os tipos. As opções são:

- **HTTP/HTTPS**;
- **Mega Upload**;
- **FTP**;
- **Disk** (um diretório local).

Para adicionar um destino do tipo Disk, é preciso ter um diretório no computador para escolher; neste exemplo vou utilizar `~/Documents`. Sempre que alterar o projeto, clique no botão **Save** — salve sempre, pois a persistência **não** é automática.

O ciclo de trabalho: exportar, importar, picotar

Agora, com o LibreOffice, vou criar um documento chamado `dossie*politicos*brazil.odt` para testes, adicionado em `~/Documents`. Depois, exporte o arquivo — clique no botão **Export** sempre que quiser salvar no servidor remoto, e **Import** quando quiser fazer o download. Veja que a Treeview passa a apresentar o arquivo. Se voltar ao diretório `~/Documents`, verá que o arquivo virou um **link simbólico (symbolic link)** — por isso é muito importante **salvar**. Volto a insistir: salve.

Depois, destrua o arquivo no seu computador local clicando no **picotador de arquivo (file shredder)**. Agora pode fechar a sua Codó GUI. Quando abrir novamente o programa, você terá de selecionar um arquivo de definição — este arquivo foi criado quando você criou o projeto. Você teve de definir um número; no caso deste material, foi `17366285713012547`, e o diretório foi `/home/usuario`, então o arquivo criado foi `17366285713012547.json` em `/home/usuario`. O objetivo de salvar o arquivo com o número, e não com o nome do workspace, é evitar que o usuário dê a pista do real motivo da existência do arquivo `.json`. Informe a chave de criptografia e clique em **Open Workspace**. Ao entrar, o primeiro passo é importar o arquivo pressionando o botão **Import**.

Configurando um serviço Apache2 na rede local ou remoto

Uma das formas mais rápidas de transferir arquivos é por meio de HTTP, e ainda, com Apache2 e PHP, consome-se pouca memória — por isso os serviços de hospedagem de PHP

são tão baratos. Projetei uma forma de armazenar em HTTP os arquivos remotos, e para isso há duas abordagens:

1. O hacker configura o seu próprio serviço HTTP com Apache2 e PHP;
2. O hacker contrata um serviço de baixo custo.

Na opção 1, o hacker pode utilizar uma VPS na Internet ou até ter o seu próprio serviço local na LAN, mas terá de manter o serviço sempre ativo. Na opção 2, basta manter o pagamento em dia que o serviço sempre estará lá — uma boa alternativa.

Configurando um serviço local

Começo demonstrando a configuração básica de um Apache2 em um Debian 12. Primeiro, é preciso ter um Debian perfeitamente atualizado e conectado à rede de computadores, de preferência na LAN, caso você queira mais privacidade. Execute os comandos abaixo para atualizar o Debian e instalar o Apache2 e o PHP:

```
  1 sudo apt update -y
167 sudo apt install apache2 php
```

A instalação é muito rápida. Para verificar se o serviço está executando, faça um restart e depois solicite um status:

```
  1 sudo systemctl restart apache2.service
168 sudo systemctl status apache2.service
```

Veja que o serviço está `enabled` (ativo para a inicialização) e também `active` (rodando perfeitamente). Digite `q` para sair. O Apache2 está associado ao usuário `www-data` e ao grupo `www-data`, e foi criado um diretório para os arquivos em `/var/www/html`. O próximo passo é criar a estrutura de que precisamos. O hacker pode criar qualquer sequência de diretórios dentro de `/var/www/html`; para este exemplo, vou colocar diretamente na raiz, deixando bem óbvio. Temos de criar um diretório `uploads`, dar a permissão para o usuário e garantir que o `www-data` seja o dono de todo o diretório `/var/www/html`:

```
  1 sudo mkdir -p /var/www/html/uploads
169 sudo chmod 777 /var/www/html/uploads
170 sudo chown -R www-data:www-data /var/www/html
```

Através do **SCP**, envie os arquivos `download.php`, `upload.php` e `status.php`, bem como o diretório `data`, que estão no projeto CodóEncrypt em `codoenc-main/http/`. Se você não sabe operar com SCP, deve aprender Linux antes de querer ser um hacker. Depois, digite o comando para garantir que o usuário do Apache seja o dono do diretório — afinal, acabamos de mover arquivos para lá e não queremos falhas por falta de permissão:

```
  1 sudo chown -R www-data:www-data /var/www/html
```

Veja que o arquivo `/var/www/html/data/config.json` possui um **token**; por padrão, deixei `123456`. Você deve mudá-lo (utilize um editor) e decorar qual é o token, pois ele será útil no próximo passo. Agora, no seu computador, sabendo que o serviço está em `http://192.168.1.137/` (no meu caso, na minha rede), informe o caminho e a chave de acesso (o token) na Codó GUI. Salve e feche: se acertou o IP e o token, o serviço será adicionado à lista. Caso queira adicionar TLS/SSL, existe documentação específica para o Apache. É altamente recomendável fazê-lo em qualquer serviço exposto à Internet.

Contratando um serviço web

Caso contrate um serviço web — como Hostgator, Hostinger, Locaweb etc. —, você deve simplesmente abrir a interface e realizar o upload de `download.php`, `status.php` e `upload.php`, bem como do diretório `data`, que estão no projeto CodóEncrypt em `codoenc-main/http/`. Da mesma forma, recomendo que troque o token de `123456` para outra sequência de caracteres. Se puder, crie diretórios internos só para dificultar o acesso direto pela raiz. Crie um novo diretório chamado `uploads` e dê permissão `777`. Deixe um `index.html` vazio tanto em `uploads` quanto na raiz — isso vai evitar a listagem de diretórios.

Criando um serviço FTP na rede

Uma boa solução, se comparada à sua segurança, é o **FTP**. O FTP com TLS/SSL é muito melhor que o serviço HTTP com TLS/SSL neste quesito, mas perde em desempenho. Às vezes temos de pagar pela nossa segurança, e tempo é uma dessas moedas. Neste exemplo, vou criar na LAN um serviço FTP sem TLS/SSL. O primeiro passo é instalar; para isso, execute os comandos abaixo:

```
  1 sudo apt update -y
171 sudo apt install vsftpd -y
```

A configuração básica refere-se à liberação de recursos, atuando sobre valores no arquivo de configuração. Recomenda-se que, antes de alterá-lo, você faça uma cópia de segurança:

```
1 sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.old
```

Agora abra com o `nano` o arquivo `/etc/vsftpd.conf` e edite cada atributo descrito abaixo:

- Altere o valor de `listen` para `YES` — isso fará com que o serviço seja inicializado a partir de um script de inicialização do sistema (init ou systemd) e também fará do processo um daemon de serviço;
- Desative o IPv6, pois o serviço estará disponível só em IPv4: passe `listen_ipv6` para `NO`;
- Por padrão, o serviço FTP vem configurado somente para leitura; remova o comentário de `write_enable`, ativando a possibilidade de escrita;
- Por padrão, vem desabilitada a opção `chroot*local*user` — este parâmetro, quando ativo, prende cada usuário ao seu diretório pessoal (**chroot jail**); caso queira permitir que, além de ver o diretório, o usuário também possa escrever nele, ative o parâmetro `allow*writeable*chroot`.

O arquivo então recebeu alterações nos seguintes valores:

```
  1 listen=YES
172 listen_ipv6=NO
173 write_enable=YES
174 chroot_local_user=YES
175 allow_writeable_chroot=YES
```

Atenção: no Debian 12, é preciso adicionar também `local_enable=YES` à listagem acima. Em seguida, execute os comandos para reiniciar o serviço e ver o status:

```
  1 sudo systemctl restart vsftpd.service
176 sudo systemctl status vsftpd.service
```

Veja que o serviço está `enabled` (será executado na inicialização do Linux) e `active` (sendo executado neste momento). Agora, pegue o endereço IP da máquina — neste caso,

`192.168.1.100`, mas isso muda de rede para rede. Abra a sua Codó GUI e configure o serviço remoto FTP. Utilize o endereço IP da máquina, o `username` e o `password` do serviço FTP. O diretório `/` significa o diretório do usuário no Linux, então os arquivos serão salvos no servidor em `/home/userlinux/`. Se você entrar com todos os dados corretos, o serviço FTP aparecerá na lista de servidores. Num teste feito com seis arquivos, tudo funcionou.

Criando uma conta no Mega Upload

O **Mega** é uma opção resiliente e externa. Por não estar na sua LAN, possibilita maior ocultação — mas deve-se utilizar uma rede I2P, Tor ou VPN. A versão gratuita dá direito a **20 GB** de dados, o que é um bom volume para quem está começando. Entre no site `mega.io` e faça o cadastro, lembrando-se de guardar a senha. Depois, entre na Codó GUI e cadastre o serviço Mega, informando apenas o `username` e o `password`. Passando pelo teste, ele estará na sua lista de servidores. Ao adicionar alguns arquivos na Codó GUI e exportar para o servidor remoto, veja que os arquivos ficam em diretórios com nomes variados e aleatórios. De todos os tipos de serviços remotos, o Mega é o mais lento, porém é o mais confiável quanto à sua existência num futuro de longo prazo e fora da sua LAN.

Apagar de verdade: anti-forense e o problema do SSD

A criptografia esconde o conteúdo, mas há um segundo front igualmente importante: **destruir vestígios**. O autor menciona o "picotador de arquivo" da Codó GUI e a destruição do arquivo local — vale entender o que "picotar" significa e por que, no hardware moderno, isso é mais difícil do que parece.

Em um disco rígido tradicional (**HDD**), apagar um arquivo pelo sistema apenas remove a entrada no índice; os dados continuam gravados nos setores até serem sobrescritos. Por isso existem utilitários de **sobrescrita segura (secure wipe)**, sendo o mais clássico o `shred`, que grava várias passagens de dados aleatórios por cima do arquivo:

```
  1 # sobrescreve 3 vezes, zera ao final (-z) e remove o arquivo (-u)
177 shred -v -n 3 -z -u arquivo_secreto.odt
179 # para um dispositivo inteiro (cuidado!)
180 sudo shred -v -n 1 /dev/sdX
```

O problema — e isto é crucial — é que **`shred` e ferramentas equivalentes não funcionam de forma confiável em SSD, cartões de memória ou pendrives**. A razão é a arquitetura da memória flash. Um SSD usa **wear-leveling (nivelamento de desgaste)**: para não gastar sempre as mesmas células, o controlador redireciona as escritas para blocos físicos diferentes, mantendo um mapa interno entre o endereço "lógico" que o sistema enxerga e o endereço "físico" real. Quando você manda o `shred` sobrescrever um arquivo, o controlador quase sempre grava as novas passagens em **outras** células físicas, deixando as originais intactas em áreas que o sistema operacional nem consegue endereçar (a **over-provisioning area**). Some-se a isso o comando **TRIM**, que apenas marca blocos como livres para a coleta de lixo interna, sem garantia de quando — ou se — serão de fato zerados.

A consequência prática é dura: em mídia flash, não confie em apagar arquivo por arquivo. As abordagens que realmente funcionam são (a) **criptografar tudo desde o início** — se o disco inteiro é LUKS e você destrói o cabeçalho/keyslots, os dados remanescentes viram lixo indecifrável, o chamado **crypto-erase**; e (b) usar o comando de **ATA Secure Erase** do próprio firmware do SSD, que instrui o controlador a apagar todas as células, inclusive as escondidas. Ou seja: no mundo dos SSDs, a melhor "borracha" é ter cifrado antes e jogar fora

a chave. Isso reforça, do ponto de vista técnico, exatamente a estratégia de camadas que o autor defende.

Onde o CodóEncrypt se posiciona: tomb e Shufflecake

Vale situar a ferramenta do autor no ecossistema, sem desmerecer nem exagerar. O CodóEncrypt tem uma proposta incomum e criativa: em vez de um único container, ele **fatia** cada arquivo, cifra cada pedaço com chave própria e **distribui** os fragmentos por vários back-ends remotos (HTTP, FTP, Mega, disco). Essa dispersão geográfica/logística é a sua grande sacada — o adversário precisa comprometer todos os destinos e ainda remontar fragmentos anônimos.

Duas ferramentas conhecidas ajudam a entender o terreno por comparação. O **tomb**, mantido pelo coletivo Dyne.org, é essencialmente um script que empacota o fluxo LUKS: cria um arquivo-container ("tomba"), guarda a chave separada (podendo escondê-la dentro de uma imagem por esteganografia) e monta/desmonta com comandos simples. É a mesma ideia do disco virtual LUKS do CodóEncrypt, porém local e sem fatiamento remoto. Já o **Shufflecake** (também da Dyne.org) é o herdeiro espiritual dos volumes ocultos do VeraCrypt para Linux: permite criar **múltiplos volumes ocultos aninhados** num mesmo dispositivo, cada um com sua senha, oferecendo negação plausível em várias camadas — você pode revelar dois ou três volumes "chamariz" e manter os demais indistinguíveis do espaço livre.

O ponto honesto é este: nenhuma dessas ferramentas, incluindo o CodóEncrypt, substitui as demais camadas — todas assumem que a criptografia de disco, a disciplina de opsec e o anonimato de rede (Tor/I2P/VPN) já estão no lugar. A força está na soma. É por isso que o capítulo começou pela defesa em profundidade, e não por uma "bala de prata".

Ferramentas citadas

- **VeraCrypt** — criação de volumes e discos criptografados com suporte a volumes ocultos (negação plausível); fork do TrueCrypt; **código aberto** (Apache 2.0 e licença TrueCrypt 3.0).
- **CodóEncrypt** — ferramenta do autor: disco virtual LUKS, cifra por arquivo, fatiamento em pedaços de nome aleatório e distribuição por servidores remotos; distribuída gratuitamente no SourceForge.
- **LUKS / cryptsetup** — padrão de cabeçalho e utilitário para criptografia de disco no Linux, sobre o dm-crypt; **código aberto** (GPLv2).
- **Apache2** — servidor HTTP usado como back-end de armazenamento remoto (`upload.php`, `download.php`, `status.php`); **código aberto** (Apache License 2.0).
- **vsftpd** — servidor FTP leve usado como back-end de armazenamento; **código aberto** (GPLv2).
- **Mega** — armazenamento em nuvem com cifragem do lado do cliente, 20 GB gratuitos; serviço **comercial** (cliente de código aberto).
- **tomb** — empacotador de volumes LUKS com chave separável; **código aberto** (GPLv3).
- **Shufflecake** — volumes ocultos aninhados com negação plausível no Linux; **código aberto** (GPLv3).
- **shred / GNU coreutils** — sobrescrita segura de arquivos em mídia magnética; **código aberto** (GPLv3).

Referências

- FRUHWIRTH, Clemens. **LUKS On-Disk Format Specification**. 2005–2018. Disponível em: https://gitlab.com/cryptsetup/cryptsetup/
- NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). **FIPS 197: Advanced Encryption Standard (AES)**. Gaithersburg, 2001 (atualização de 2023).
- VERACRYPT. **VeraCrypt Documentation — Plausible Deniability e Hidden Volume**. Disponível em: https://www.veracrypt.fr/
- KISSEL, Richard et al. **NIST SP 800-88 Rev. 1: Guidelines for Media Sanitization**. NIST, 2014.
- DYNE.ORG. **Tomb: the Crypto Undertaker** e **Shufflecake: plausible deniability for multiple hidden filesystems**. Disponível em: https://dyne.org/
- ANDERSON, Ross. **Security Engineering**. 3. ed. Wiley, 2020 (capítulos sobre criptografia de disco e anti-forense).

Gravação de Vídeo Anônima

Uma ação muito comum no hacktivismo são gravações, sejam comunicados de ações ou até mesmo a passagem de conhecimento hacker. Acontece que neste ponto muitos hackers se revelam — seja por uma tosse constante, seja por um som de fundo. Neste capítulo vou trazer a experiência de anos nesta área, e vou além: cada cuidado de estúdio que descrevo tem, hoje, uma pesquisa acadêmica ou um caso real por trás explicando por que ele importa. Anonimizar um vídeo não é editar um efeito de voz e postar; é fechar, um a um, os canais laterais pelos quais o ambiente, o corpo e a própria máquina denunciam quem está atrás da máscara.

Preparando o ambiente de gravação

Um passo muito importante. Vários elementos podem lhe associar a um vídeo, e o objetivo da montagem deste ambiente é desconectar você do cenário real e evitar algo pior que a associação, que é a **localização** do hacker. Veja que no caso do seriado *Mr. Robot* houve a variação de ambiente em dois momentos, mas o ambiente inicial era perto do mar — ou seja, gaivotas e barcos lançam sinalizadores sonoros indiscutíveis naquele cenário. Já na segunda cena (a casa da empresária), veja que não houve a preparação do ambiente de forma adequada. A ficção erra de propósito; nós não podemos errar.

O primeiro passo é o cenário. O isolamento deve ser ao máximo. O ideal é construir um estúdio no interior de uma sala, mas **não apoiar o estúdio nas paredes**, e na parte inferior colocar uma grande borracha — a mesma usada em borracharias. Todo o chão do estúdio, bem como o próprio estúdio, deve estar em cima de placas de borracha para evitar a trepidação de uma avenida no som da gravação. Geralmente estas placas de borracha são vendidas para borracharias, e é ali que você as consegue sem levantar suspeita.

Tudo deve estar aterrado. A mesa deve ter fitas metálicas ligadas à fonte, para que o hacker dissipe a energia estática — **eu já usei uma fita metálica perto do teclado**. Esse aterramento tem duas funções: proteger o hacker e, principalmente, reduzir o zumbido elétrico (o *hum* de 60 Hz) que a rede injeta no áudio e que, sozinho, já pode denunciar a frequência da rede elétrica do seu país.

Distâncias e o chroma key

Atrás da cadeira, na parede que vou chamar de PAREDE 1, fica o fundo. Para a iluminação ser perfeita e o **chroma key** ficar perfeito, deve-se ter no mínimo 3 metros de parede, e entre a cadeira e esta parede no mínimo 1.8 metro. O ideal é que este interior tenha no mínimo 3x3 metros. Se o chroma key apresentar alguns erros pontuais, será então possível ligar o hacker ao vídeo — ou seja, o chroma key é imprescindível para tirar o fundo e desacoplar o vídeo do local de gravação. Já vi estúdios com dimensões inferiores a 3x3 nos quais a falha de iluminação no fundo verde deixava aparecer a quina de uma parede ou a sombra do corpo, e essa sombra é geometria: ela entrega o pé-direito e o tamanho da sala.

O teclado, o corpo e o celular

Tanto a mesa como o teclado (**keyboard**) devem ser silenciosos. Geralmente teclados de mola são tão barulhentos quanto gaivotas e barcos; faça um emborrachamento da mesa e consiga um teclado silencioso. A forma como você digita também pode ser utilizada contra você. E

aqui vale um aprofundamento que reforça, com pesquisa séria, a recomendação do teclado silencioso: existe toda uma linha de estudos sobre **emanações acústicas de teclado** (*acoustic keyboard emanations*). Desde o trabalho de Dmitri Asonov e Rakesh Agrawal, na IBM (2004), demonstrou-se que **cada tecla produz um som ligeiramente diferente**, e que é possível reconstruir o que foi digitado apenas gravando o clique das teclas. Trabalhos posteriores, inclusive com técnicas de aprendizado de máquina, elevaram muito a taxa de acerto. Ou seja: um teclado audível no seu vídeo não é só um ruído incômodo — é um vazamento de dados. Se você digita uma senha, um comando ou uma URL enquanto grava, esse áudio pode, em tese, ser transcrito por quem analisa o material. Silencie o teclado, emborrache a mesa, e — melhor ainda — não digite nada sensível com o microfone aberto.

Outro elemento barulhento, e que pode levar a máfia até o seu local, é o celular. O celular emite sons, e pelo momento em que toca e pela diferença de tempo entre os toques (WhatsApp, Telegram, etc.) dá para realizar um filtro sobre a conta telefônica, e naturalmente, por **triangulação**, chegar até a sua localização. **Evite celular no estúdio** — desligado não basta; deixe-o fora, longe, de preferência em outro cômodo ou fora de casa.

O som de fundo que embaralha

Outro som que pode entregar a posição é o som de fundo. No caso de *Mr. Robot*, deveria haver a proximidade com o mar. Uma internet boa, com boa capacidade elétrica, restringe bastante as áreas de busca; então posicione **duas caixas de som INDEPENDENTES**, e principalmente independentes de CELULAR e de COMPUTADOR. Consiga sons de fundo de locais distantes: se você está no interior, evite galinhas e consiga sons de rodovias, portos, proximidade com o mar, avenidas movimentadas. Esse som de fundo é o que vai embaralhar; vai ficar impossível localizar o local da gravação pelos sons da sua localidade.

Mas qual o motivo de as caixas serem independentes? Simples: ambas devem ter timbres diferentes para gerar muitos ruídos de diferentes frequências e tons. Duas fontes independentes, com timbres distintos, criam um campo sonoro que não se cancela nem se sincroniza — e isso dificulta que um analista subtraia o "ruído de fundo" para revelar o ambiente real por baixo. Se as caixas saíssem do mesmo computador ou do mesmo celular, um adversário poderia identificá-las como uma única trilha somada e removê-la.

Comprando sem deixar rastro

Agora, como comprar estes itens? Se foi pela internet com o seu cartão de crédito, digo que você acaba de estar associado a um possível estúdio de gravação! Mesmo que consiga montar um estúdio sem se associar aos produtos, muitos hackers compram máscaras — e com as máscaras ficam associados. Mas existe uma saída: há máscaras sem desenhos ou cores, e então o hacker pode pintar com tinta guache ou até mesmo com **papel machê** envenenar o *look* da máscara, criando uma peça única que não existe em nenhum catálogo. Há a possibilidade ainda de usar uma camisa que não deixaria rastro (se não tiver estampa) ou uma touca ninja adquirida com dinheiro em alguma loja de motoqueiros — sim, os motoqueiros a usam por baixo do capacete.

Não deixe rastros de cartão de crédito, nem em sites como o Mercado Livre. E cuidado com a própria tela: os **ícones da área de trabalho** podem entregar o que o hacker possui, e pelo que ele possui dá para chegar aonde ele vai. Procure eliminar todos os ícones e também trocar o papel de parede. Evite gravar a barra superior com o horário de gravação e com a localização. Um relógio de sistema com fuso horário, um ícone de um aplicativo regional, o nome de uma rede Wi-Fi na bandeja — tudo isso é metadado visual.

Reflexos: o inimigo que você não vê

Há um canal lateral que quase ninguém trata e que já desmascarou gente: os **reflexos**. Pesquisadores demonstraram que é possível reconstruir aquilo que uma pessoa está vendo — e o ambiente ao redor — a partir do reflexo na **córnea** dos olhos em fotos de alta resolução (o trabalho de Nishino e Nayar, e mais tarde estudos que reconstruíram telas de computador a partir do reflexo em óculos e em objetos brilhantes na cena). Traduzindo para o nosso estúdio: se você usa óculos, a lente pode refletir o monitor, a janela, a disposição do cômodo. A córnea faz o mesmo. Uma caneca, um copo, a superfície de um gabinete polido, o vidro de um quadro na parede — qualquer superfície especular na cena pode conter uma miniatura do que você quer esconder. Antes de gravar, examine o quadro procurando brilhos, e elimine superfícies reflexivas do campo de visão. Se usa óculos, considere hastes e lentes que não devolvam a imagem da tela, ou reposicione a iluminação para matar o reflexo.

Ferramenta de gravação

Duas ferramentas são ótimas para realizar a gravação. Na lista abaixo, elas aparecem na ordem de complexidade e também de recursos:

1. **SimpleScreenRecorder** (SSR);
2. **OBS Studio**.

O SSR é muito simples, e geralmente já vem instalado em GNU/Linux de alto nível. Com quatro telas simples (geralmente já vem configurado), o hacker pode iniciar a gravação da sua *screen* e, naturalmente, expor a sua informação. O arquivo gerado é bruto — ou seja, com todo o ruído de fundo e com a sua screen como ela está —, mas é um arquivo considerado pequeno.

Outro poderoso aplicativo é o OBS Studio, porém este é mais complexo e indicado para quem já está atuando há um tempo, com a possibilidade de montar cenários e aplicar efeitos já na gravação do vídeo. Rapidamente o usuário pode trocar o cenário apenas clicando sobre o cenário desejado; cada cenário pode conter elementos, tais como fundo, *picture-in-picture*, vídeos, páginas web, etc.

Removendo o noise na gravação

A primeira ação a se tomar é remover o **noise**. Noise é um ruído de fundo que pode ser proveniente de:

- do seu próprio hardware (mesmo o computador estando externo ao estúdio);
- da sua respiração;
- de algum detalhe do ambiente interno e externo.

Selecione o dispositivo de áudio, clique com o botão do mouse em **filtros**, adicione o filtro de *noise* e configure a sensibilidade do efeito. Evite cortar demais, pois, quando falamos, iniciamos em uma entonação mais baixa e partimos para a mais alta; então, se o noise estiver cortando demais, parte da sua fala será cortada. Nunca dá para ser preciso, mas repare que a perda de um trecho da fala é insignificante quando bem configurado. Essa configuração vai variar de ambiente para ambiente, então é relativo. O OBS vem com uma configuração padrão que pode não ser o que você precisa; sempre teste, teste e teste mais. Um problema é quando o ruído está acima do início da sua fala: aí o corte vai acabar perdendo parte da sua fala, ou, ao iniciar a fala, aparece o ruído de fundo.

Às vezes o *record* da screen não é tão perfeito e acaba mostrando uma pequena linha nas bordas — como nas ondas gravitacionais, podemos deduzir o que está por baixo da janela que está sendo gravada. Recomenda-se adicionar ao cenário um fundo de uma cor; utilize **PRETO**, pois preto cai bem em qualquer look.

Evite o uso de HD (disco) e mantenha o computador fora do estúdio, geralmente bem aterrado, para evitar interferência no áudio — aterramento só para o computador, e não urine no seu aterramento. Esqueça o uso de notebooks para esta atividade e, se puder, coloque *water cooler* no PC, com o resfriador externo, para que nem o barulho da ventoinha entre no microfone.

Antes de iniciar a gravação, respire e pense bem na introdução. Quando iniciar a gravação, fique 30 segundos com a sua respiração normal; não fique eufórico e não prenda a respiração, pois vai lhe faltar ar nos próximos 2 minutos. Esses primeiros segundos parados não são desperdício — como veremos, eles são a matéria-prima para o Audacity aprender o "silêncio" do seu ambiente.

Ferramenta de edição de voz

Mesmo com todo o cuidado, haverá a necessidade de editar o áudio, e a ferramenta indicada é o **Audacity**. Além de trabalhar a qualidade do áudio, é fundamental aplicar um efeito para desfocar a voz. Para iniciar, clique com o botão direito do mouse sobre o arquivo gerado na gravação do OBS Studio e abra com o Audacity.

O primeiro passo é pegar um padrão de noise do ambiente e, em especial, da sua respiração. Se o ambiente foi montado como descrito aqui, o único ruído será você. Então selecione os **20 primeiros segundos** do seu vídeo e extraia um padrão de noise (em *Effect → Noise Reduction → Get Noise Profile*). Agora selecione todo o vídeo para aplicar a redução com base nesse perfil.

A segunda ação sobre o áudio é aplicar efeitos para garantir que a voz não seja explicitamente associada ao hacker. Um efeito simples e que garante um bom resultado é o **Pitch**: com todo o vídeo selecionado, acesse o efeito Pitch, altere os valores e a frequência alterada. Teste a frequência entre **-18 e -25**, use o *Preview* para avaliar e dê OK se gostar da voz. Pode explorar mais efeitos. Para exportar é fácil: em *File*, selecione *Export* e exporte para mp3.

Por que só mudar o pitch não basta

Aqui preciso ir além do que eu fazia. O *pitch shift* simples — subir ou descer o tom da voz — é a técnica mais comum, mas é também a mais **reversível**. O motivo está na física da voz. O tom (a frequência fundamental, o *pitch*) é apenas uma das camadas do sinal; a identidade da sua voz está principalmente nas **formantes** (*formants*), que são as ressonâncias do seu trato vocal — o formato da sua boca, garganta e cavidade nasal. Quando você aplica só um pitch shift, dependendo do algoritmo, as formantes ficam preservadas ou apenas deslocadas de forma proporcional e previsível. Isso significa que um analista pode aplicar o deslocamento **inverso** e recuperar algo muito próximo da sua voz original, ou comparar as formantes com uma amostra conhecida e casar a identidade — porque o padrão formântico é uma espécie de impressão digital do seu aparelho fonador.

A recomendação, portanto, é não parar no pitch. Altere também as **formantes** (o Audacity tem *effects* que fazem *formant shifting*, e ferramentas como o `rubberband` permitem separar o deslocamento de pitch do deslocamento de formante). O ideal é combinar: deslocar

o pitch, deslocar as formantes em proporção diferente, e ainda somar leve alteração de tempo e alguma coloração. Quanto mais você quebra a relação natural entre pitch e formantes, mais difícil fica reverter para a voz original e mais fraca fica qualquer tentativa de comparação biométrica. O pitch entre -18 e -25 continua sendo um bom começo — só não pode ser o fim.

Metadados e a impressão digital do arquivo

Antes de dar por concluído o áudio (e depois o vídeo), há um cuidado que não é sobre o que se ouve, e sim sobre o que está **escrito dentro do arquivo**: os **metadados**. Todo arquivo de mídia carrega campos ocultos — no áudio, tags ID3 e campos de codificação; no vídeo, os **metadados EXIF** e de contêiner (data e hora exatas, fuso horário, modelo do software de codificação, às vezes GPS quando a fonte é um celular, o *encoder* usado). Um vídeo exportado com data, hora e fuso preenchidos entrega o momento e a região da gravação sem que uma única palavra tenha sido dita. Limpe tudo antes de publicar. No Linux, ferramentas como `exiftool` e `ffmpeg` (com `-map_metadata -1`) removem esses campos.

Só que remover os metadados textuais não apaga tudo. Existe uma assinatura mais profunda: o **fingerprint do encoder** e, quando há câmera envolvida, o **PRNU** (*Photo-Response Non-Uniformity*), o ruído próprio do sensor. Cada codificador de vídeo deixa padrões característicos (a forma como quantiza, os artefatos de compressão), e cada sensor de câmera imprime, em cada quadro, um padrão de ruído fixo e único — imperceptível a olho nu, mas estatisticamente estável, a ponto de a perícia usar o PRNU para **ligar um vídeo a uma câmera específica**, do mesmo jeito que a balística liga uma bala a um cano. Isso importa para o hacker de duas formas: primeiro, se você grava com câmera (e não só a screen), o sensor daquela câmera pode ligar este vídeo a outros vídeos seus; segundo, reencodar o material com parâmetros padronizados e comuns, e evitar reutilizar a mesma câmera física entre operações, ajuda a diluir essas assinaturas. Não confie em "apaguei os metadados" como se fosse anonimato completo — é apenas a primeira camada.

Ferramenta de edição de vídeo

Uma ótima ferramenta é o **Kdenlive**, que roda muito bem no GNOME apesar do nome. Além de um editor de trilhas, permite:

- adicionar efeitos visuais;
- adicionar efeitos de áudio;
- edição de elementos.

Arraste o vídeo original gerado no OBS Studio e o mp3 gerado no Audacity para o projeto. Agora que ambos os elementos estão no projeto, é só arrastar o vídeo para a primeira trilha (automaticamente o áudio será jogado na terceira trilha) e jogar o mp3 gerado no Audacity na segunda trilha. O próximo passo é inibir a trilha A2, para não ter dois áudios: basta clicar sobre o símbolo de áudio da trilha e colocá-la como mudo. Depois, remova o início — os 20 a 30 segundos iniciais que foram gravados para ajustar o mecanismo de noise no Audacity. Seu vídeo está pronto para ser renderizado, mas **salve antes!!!** E, na hora de renderizar, escolha um perfil de saída comum e limpe os metadados, como já dito.

Hardware de mixagem de voz

Um problema de uma mixagem via software é que é possível analisar o áudio conhecendo o software utilizado. Uma boa ideia (não tenho grana para ter, então não testei) é utilizar a alteração da voz por **hardware**, antes do *input* do computador, e assim até impedir a intrusão

e o roubo do áudio original. A lógica é forte: se a voz já entra modificada na máquina, um malware que grave o microfone jamais captura a sua voz real, porque ela nunca chega a existir em formato digital dentro do computador.

Um bom microfone — aprendi que os microfones decentes são da **Behringer** — e a entrada do microfone vai no **Vocal Performer**, que é uma pedaleira de voz que permite aplicar efeitos, inclusive o Pitch. Caso queira aumentar o ganho, pode-se ainda adquirir um **compressor**. Recomenda-se o uso de uma placa **Sound Blaster** como placa de áudio *off-board*: esta placa não sofre interferência da placa-mãe. Com este equipamento, o hacker estará mais seguro e poderá inclusive participar de *lives* ou discussões em grupo por voz.

Roteiro e palavras reservadas

No Brasil temos algumas particularidades regionais. Por exemplo, no estado de Minas Gerais usa-se o termo "**trem**" para tudo: um carro é um trem, um computador é um trem, uma chave de fenda é um trem. Se isso é usado em um texto dito ou redigido pelo hacker, fica fácil saber e restringir pessoas em uma busca. Então, se o roteiro tiver termos regionais, eles devem ser usados **para despistar** possível intervenção mafiosa — plante propositalmente marcas de uma região que não é a sua. Isso vale para a **estilometria** de forma geral: o vocabulário, os bordões, os erros recorrentes e o ritmo de fala são uma assinatura tão pessoal quanto a voz. Crie um roteiro bem pensado e depois treine, treine bem. Fale devagar e pense nas próximas palavras. Não se entregue.

Modificando a voz em tempo real com Clownfish for Linux

O **Clownfish Voice Changer** é um software de modificação de voz para Windows, GNU/Linux e Mac, capaz de converter o seu áudio em qualquer outra voz de uma lista pré-definida; há também versões para Android e iOS. É útil para quem cria conteúdo de áudio e quer alterar a voz em **tempo real**, funcionando com plataformas como Discord, YouTube e Odysee. É um dos aplicativos de alteração de voz mais populares, muito usado por *YouTubers* e *gamers* para alterar a voz ao vivo em transmissões, bate-papos por voz e jogos. Comparado a alternativas como Voicemod Pro ou Morphox, uma vantagem do Clownfish é o suporte multiplataforma e o fato de ser gratuito — os concorrentes costumam ser só para Windows e pagos.

O aplicativo é instalado no dispositivo e só precisa acessar o microfone para alterar a voz em tempo real. Algumas das vozes da ferramenta são: voz alienígena, Atari, clone, mutação, mutação rápida, mutação lenta, campo masculino, campo feminino, passo de hélio, campo para bebês, rádio, voz de robô, voz AI, silêncio e proposta personalizada.

Para fazer o download, entre no site oficial e, na seção de *Download*, escolha a opção **CFConsole**, pois queremos usar no GNU/Linux de forma não gráfica. O download é rápido; descompacte o arquivo `.zip` e, pelo console, navegue até o local — o executável `ClownfishConsole` já vem com permissão de execução. O programa é muito simples: *Turn on/off* liga ou desliga o efeito; para configurar em *Audio Settings*, é preciso desligar (OFF) o efeito. Na parte inferior é possível ver qual placa de som entra o áudio e qual placa de som sai o áudio; o **Built-in Audio Digital Stereo** é uma placa virtual interna, e é esta placa que deve ser selecionada no OBS Studio. Após configurar a placa de som, o hacker deve entrar e selecionar a voz — a lista é grande, basta selecionar. Embora seja digital, saiba que **pode falhar**; teste antes de qualquer transmissão ao vivo, porque uma falha do efeito no meio de uma live devolve a sua voz real ao mundo.

Sistema de exaustão de ar em estúdios

Com certeza uma boa cerveja e um ar fresco ajudam a criar bons vídeos. No seu estúdio, com tanta espuma e tanta borracha, é fundamental a **exaustão de ar** e a renovação do oxigênio. Com uma caixa de madeira, criando uma espécie de labirinto para o ar, é possível abafar o ruído externo enquanto o ar circula. Recomendo que faça com **madeirite**, e maximize de acordo com o tamanho do estúdio. No meu caso, utilizo um com passagem de **10 cm x 10 cm**, e consigo uma boa exaustão de ar quente. O labirinto é o segredo: o ar entra e sai, mas o som, obrigado a fazer as curvas do caminho, perde energia antes de chegar ao microfone — e você não desmaia dentro de uma caixa forrada de borracha.

Ferramentas citadas

- **SimpleScreenRecorder (SSR)** — gravador de tela simples e leve para GNU/Linux; **código aberto** (GPL).
- **OBS Studio** — estúdio de gravação e transmissão com cenários, filtros e efeitos em tempo real; **código aberto** (GPL).
- **Audacity** — editor de áudio multifaixa, usado aqui para redução de noise e alteração de pitch/formantes; **código aberto** (GPL).
- **Kdenlive** — editor de vídeo não linear baseado no MLT; **código aberto** (GPL).
- **Clownfish Voice Changer** — modificador de voz em tempo real, multiplataforma; **gratuito** (freeware, proprietário).
- **exiftool / ffmpeg** — utilitários para inspeção e remoção de metadados de mídia; **código aberto** (Perl Artistic License / LGPL-GPL).

Referências

- ASONOV, Dmitri; AGRAWAL, Rakesh. **Keyboard Acoustic Emanations**. IEEE Symposium on Security and Privacy, 2004.
- ZHUANG, Li; ZHOU, Feng; TYGAR, J. D. **Keyboard Acoustic Emanations Revisited**. ACM Transactions on Information and System Security, 2009.
- BACKES, Michael; DÜRMUTH, Markus; UNRUH, Dominique. **Compromising Reflections — or — How to Read LCD Monitors Around the Corner**. IEEE Symposium on Security and Privacy, 2008.
- NISHINO, Ko; NAYAR, Shree K. **Corneal Imaging System: Environment from Eyes**. International Journal of Computer Vision, 2006.
- LUKÁŠ, Jan; FRIDRICH, Jessica; GOLJAN, Miroslav. **Digital Camera Identification from Sensor Pattern Noise (PRNU)**. IEEE Transactions on Information Forensics and Security, 2006.

Sala Cofre e Defesa Física

Muitos hackers acabam ficando mais arrogantes que o próprio estado (governo), e passam a ter uma vida de relaxamento. Quantos hackers famosos foram presos com seus computadores ligados? Veja o caso de Ross William Ulbricht e outros: saiba que o governo tem meios para chegar até você, e o que você pode fazer é dificultar esse cenário — mas esteja preparado. A **sala cofre** é uma forma de você dificultar tanto a forense que o advogado terá todo o tempo necessário para lhe tirar da enrascada. E, repito, esteja preparado.

O que é a sala cofre

Trata-se de uma sala reservada para computadores preparados para serem capturados. Então esses computadores devem estar segmentados da rede, ter o disco criptografado e ter serviços que, ao serem acessados, desligam o computador. Se alguém entrar nesta sala, tudo deve ser desligado.

A regra é montar a sala com os computadores no fundo e adicionar **4 sensores de presença** nas paredes, ligados a relés em série. E aqui vem o detalhe que muita gente inverte: você deve **facilitar** a entrada do governo. Na porta, deixe a chave e coloque uma placa "Sala de Servidores". A ideia é ter um ambiente convidativo para que abram a porta e entrem — os sensores irão perceber e desligar os computadores. Você não está trancando o cofre; está armando uma armadilha que dispara quando o cofre é aberto.

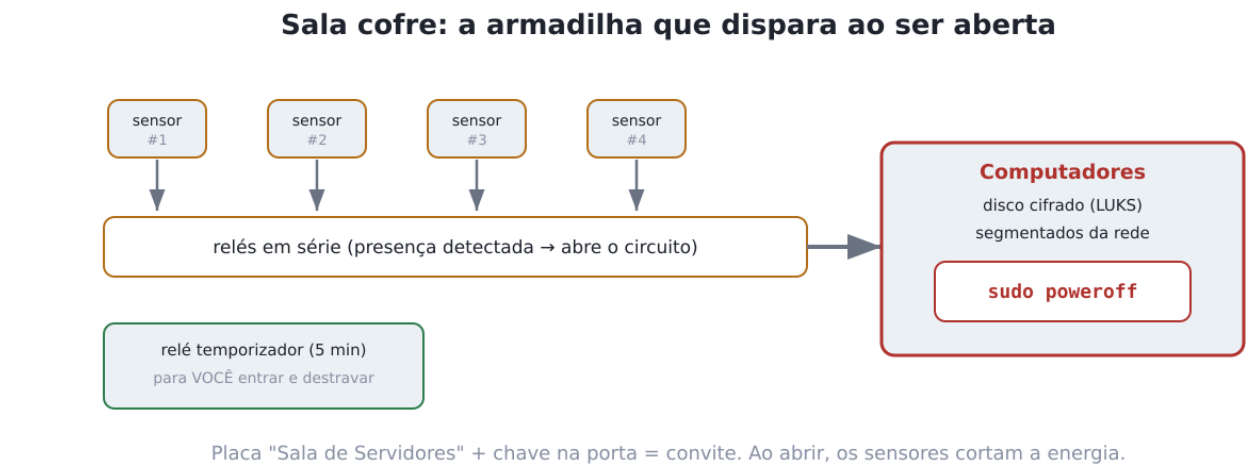

Figura: Circuito da sala cofre — sensores de presença → relés → desligamento.

As duas forenses

Existem duas forenses (assim as classifico): a realizada com a memória carregada de dados — ou seja, o computador ligado — e a forense com o computador desligado (pós-morte), onde a análise será basicamente no disco, que com certeza estará criptografado com **AES** (usando **LUKS**). É essa distinção que sustenta todo o projeto da sala cofre. Com a máquina ligada, a chave de decifração está na memória, exposta; com a máquina desligada, o perito herda apenas um bloco de bytes embaralhados que ele não consegue ler.

Para entender por que **desligar** é a jogada certa — e não apenas "bloquear a tela" ou "hibernar" —, é preciso conhecer o **ataque de partida a frio** (**cold boot attack**). Quando um disco é cifrado com LUKS, a chave que abre o volume não fica no disco: ela é derivada da sua

senha no momento em que você destrava o volume e, a partir daí, vive na **memória RAM** enquanto a máquina está ligada. LUKS, portanto, só protege de verdade a máquina **desligada**. Com a máquina ligada (mesmo com a tela bloqueada), a chave-mestra está lá, na RAM, à espera de quem souber onde procurar.

O problema, revelado no trabalho clássico de Halderman e colegas ("Lest We Remember: Cold Boot Attacks on Encryption Keys", 2008), é que a RAM **não esquece instantaneamente** quando a energia é cortada: os dados persistem por **segundos** e, se os módulos forem resfriados (os pesquisadores usaram spray de ar comprimido invertido para congelar os chips), por **minutos**. Um perito bem treinado pode desligar a máquina, remontar a memória em outro equipamento ou dar boot num sistema mínimo e **varrer a RAM atrás da chave de decifração** antes que ela se apague. É por isso que um `poweroff` limpo é infinitamente melhor do que arrancar a tomada: encerrar o sistema corretamente dá chance ao próprio sistema de descarregar e sobrescrever as chaves da memória, enquanto o corte bruto de energia é justamente a condição que o cold boot attack explora. A sala cofre desliga a máquina de propósito, e a criptografia LUKS só cumpre sua promessa porque a chave deixou a RAM.

O timer de manutenção e evitar plataformas hackeáveis

Se precisar dar manutenção, terá que desligar tudo e religar tudo — e, para isso, será adicionado um **timer de 5 minutos** (mais adiante detalho o relé temporizador). Uma observação de fundo, que vale para todo o hardware desta sala: **evite usar Arduino ou Raspberry Pi**, ambos são hackeados. São plataformas programáveis e, portanto, plataformas que um adversário pode subverter. Aqui a burrice do relé é uma virtude: quanto mais simples e menos programável for o circuito, menos superfície ele oferece.

Vou exemplificar com um ambiente montado com servidores em mini-PC. É importante notar que o governo pode perceber que você tem servidores pelo consumo de link e de energia — então evite servidores que despendem consumo elevado de energia. O mini-PC ajuda também nisso.

Mini-PCs utilizam **DC de 12V**, e por isso vou exemplificar com **conectores J4**, tanto para a entrada quanto para a saída. Com a própria entrada, alimente o sistema — isso evita que um hacker físico plugue uma fonte secundária para manter tudo ligado enquanto corta o seu circuito. Ao passar pela lógica de relés, a tensão irá decair de 12V para cerca de **11.8–11.9V**, o que é normal e esperado.

O relé é simples: ele não tem programação, e deve estar **sempre ligado** (energizado). Caso um sensor detecte movimento, ele **desarma** — acionando o relé e desligando a fonte do computador. Neste exemplo estou mostrando apenas 1 sensor, mas isso pode ser expandido para **4 sensores** e uma **placa de relé de 4 elementos**: basta ligá-los em série, de modo que qualquer um dos sensores, ao disparar, quebre a corrente e derrube a alimentação de todas as máquinas.

Como o próprio operador entra: o relé temporizador

Mas temos um problema: como entrar para fazer o setup, se o sensor vai lhe capturar? Aí entra o **relé temporizador**. Para entrar na sala cofre, você deverá acionar o mecanismo, e todos os computadores serão desligados — é inevitável, até para você. Não há exceção "de dono"; a armadilha é honesta com todo mundo.

Então você terá, fisicamente, acesso ao botão de um **relé secundário**. Este relé é um temporizador, que pode usar como base **5 minutos**. Ou seja: você terá 5 minutos para destravar o LUKS, fazer a configuração e sair da sala antes que a proteção rearme. É tempo suficiente para um operador que sabe o que veio fazer, e curto demais para quem está improvisando.

Fisicamente, você deve proteger o botão. Para isso, coloque-o dentro de uma garrafa — na verdade, **todo o circuito deve ser protegido em uma case** —, para evitar um hack físico por meio de um buraco na parede. E chegamos a um problema mais fino: imagine um microfuro na parede e o emprego de uma **micro-sonda robótica**, capaz de alcançar o botão ou os contatos por dentro. Para isso, você irá preparar a parede — e, se não quiser preparar a parede inteira, pode ser apenas uma caixa que proteja todos os equipamentos e relés.

A blindagem sugerida combina **chapa de zinco** com **vidro temperado**. A chapa de zinco, quando cortada com auxílio de óxidos ou eletrodos, acaba emitindo muita luz — o que seria captado pelos sensores. E, em caso de corte da parede seguido do corte desta estrutura por um disco, o atacante terá problemas em atravessar o vidro temperado colado à chapa de zinco. É uma barreira que, ao ser violada, **denuncia a violação** (o clarão dispara os sensores) e, ao mesmo tempo, resiste fisicamente à passagem.

Complementos ao switch físico: dead man's switch

O mecanismo do operador que descrevi — acionar para entrar, e o rearme automático — é uma variante física de um conceito que vale nomear: o **interruptor do homem morto** (**dead man's switch**). A ideia é que a proteção dispare justamente quando o controle é **perdido**: se você for arrancado da cadeira, algemado ou afastado do equipamento, o sistema desliga sozinho, sem depender de você lembrar de apertar nada no susto.

No mundo do software isso é trivial de complementar. Ferramentas como **usbkill** e **BusKill** implementam exatamente isso: um pendrive (ou um cabo magnético preso ao seu pulso, no caso do BusKill) fica plugado na máquina; no instante em que ele é **removido** — porque alguém puxou o notebook, ou porque você levantou às pressas —, o sistema executa uma ação de emergência: desligar, hibernar apagando as chaves, ou disparar um `poweroff`. Casado com o cold boot attack que vimos, o objetivo é sempre o mesmo: **tirar a chave da RAM antes que o adversário ponha a mão no hardware ligado.** A sala cofre resolve isso pela presença (sensores); o dead man's switch em software resolve pela conexão física de um dispositivo — e um complementa o outro.

Vale também citar, do lado do adversário, dois ataques que justificam o cuidado com o hardware desligado e "esquecido". O **evil maid attack** ("ataque da camareira") é aquele em que alguém com acesso físico à máquina desligada — a camareira do hotel, o agente que entra na ausência do dono — adultera o bootloader ou o firmware para capturar sua senha na próxima vez que você ligar, contornando a criptografia de disco por completo. A defesa moderna passa por **TPM** e **Secure Boot**: o TPM (um chip que guarda medidas de integridade do processo de boot) e o Secure Boot (que só permite carregar código assinado) dificultam essa adulteração, atrelando a liberação da chave a um boot não modificado. Não é infalível, mas é a razão de o hardware moderno mover a confiança para dentro do silício.

O serviço falso: um honeypot que desliga a máquina

Fisicamente você está protegido, mas ainda falta a camada lógica. Primeiro: quanto mais você evitar acesso aos serviços pela rede, melhor. Mas precisamos fazer o governo cair em **outra**

armadilha — um serviço falso que, ao ser acionado, deve desligar o hardware.

Antes do código, vale nomear a técnica, porque ela tem nome e literatura própria. O que se monta aqui é um **honeypot** (pote de mel): um serviço que existe **só para ser atacado**, sem nenhuma função legítima. Ninguém real deveria se conectar a ele — logo, qualquer conexão é, por definição, hostil. Em vez de apenas registrar o atacante (como faz um honeypot clássico), este vai além e se comporta como um **tarpit** ("poço de piche"): a armadilha reage ao contato. No nosso caso, a reação é a mais drástica possível — o serviço aparenta ser um SSH comum, o atacante tenta autenticar, e o simples ato de tentar login dispara o `sudo poweroff`, cumprindo a doutrina das "duas forenses": desliga a máquina, tira a chave da RAM e joga o adversário para a forense pós-morte, contra um disco cifrado com LUKS que ele não consegue abrir.

A implementação abaixo é parte do projeto **KFishmonger** (do autor), aqui alterada para desligar o computador. Ela sobe um serviço SSH semelhante ao OpenSSH-Server, usando a biblioteca **paramiko**, que ao ser acessado executa o comando `sudo poweroff`:

```
  1 import os, sys, threading, traceback, paramiko, socket, socketserver, inspect
181 from binascii import hexlify
183 PORT = 22
184 RETURN_MESSAGE = None
185 PATH_RSA_PRIVATE = "/var/fake/rsa_private.txt"
187 if not os.path.exists(PATH_RSA_PRIVATE):
188     with open(PATH_RSA_PRIVATE, "w") as f:
189         f.write(rsa.make()[0])
190 host_key = paramiko.RSAKey(filename=PATH_RSA_PRIVATE)
193 class Server(paramiko.ServerInterface):
194     def __init__(self, client_address):
195         self.event = threading.Event()
196         self.client_address = client_address
198     def check_channel_request(self, kind, chanid):
199         if kind == 'session':
200             return paramiko.OPEN_SUCCEEDED
201         return paramiko.OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED
203     def check_auth_password(self, username, password):
204         print('IP: %s, User: %s, Password: %s' % (self.client_address[0], username,
205                                                   password))
206         os.system("sudo poweroff")
207         return paramiko.AUTH_FAILED  # sempre dropar a negociação USUARIO/SENHA
209     def check_channel_shell_request(self, channel):
210         self.event.set()
211         return True
213     def check_channel_pty_request(self, channel, term, width, height, pixelwidth,
214                                   pixelheight, modes):
215         return True
218 class SSHHandler(socketserver.StreamRequestHandler):
219     def handle(self):
220         try:
221             t = paramiko.Transport(self.connection)
```

```
            t.add_server_key(host_key)
            server = Server(self.client_address)
            try:
                t.start_server(server=server)
            except paramiko.SSHException:
                print('Falha de negociação SSH.')
                return
            chan = t.accept(20)
            if chan is None:
                t.close()
                return
            server.event.wait(10)
            if not server.event.is_set():
                t.close()
                return
            if RETURN_MESSAGE is not None:
                chan.send(RETURN_MESSAGE)
            chan.close()
        except Exception as e:
            print('*** Caught exception: ' + str(e.__class__) + ': ' + str(e))
            traceback.print_exc()
        finally:
            try:
                t.close()
            except:
                pass

sshserver = socketserver.ThreadingTCPServer(("0.0.0.0", PORT), SSHHandler)
sshserver.serve_forever()
```

O coração do truque está no método `check*auth*password`: ele imprime o IP, usuário e senha que o atacante tentou (bom para o registro), executa `os.system("sudo poweroff")` e devolve `paramiko.AUTH_FAILED`, dropando sempre a negociação de usuário e senha. Basicamente, é criado um serviço SSH semelhante ao OpenSSH-Server que, ao ser acessado, executa o comando `sudo poweroff`. Então, se não entrarem fisicamente na sala cofre, vão lhe perguntar a senha do servidor. Você poderá informar `123456`, dar uma risadinha e disfarçar — ao tentar acessar o serviço SSH, vão falhar, e a máquina se desliga na cara deles. A senha "certa" é o gatilho da armadilha, não a chave da porta.

Ferramentas citadas

- **KFishmonger** — projeto do autor; conjunto de serviços "isca" (honeypot/tarpit) que simulam serviços reais como o SSH. Nesta variante, ao receber uma tentativa de login, executa `sudo poweroff`. Código próprio.
- **paramiko** — biblioteca Python que implementa o protocolo SSH2 (cliente e servidor); usada aqui para forjar o serviço SSH falso. **Código aberto** (LGPL 2.1).
- **cryptsetup / LUKS** — ferramenta e formato padrão de criptografia de disco no Linux (LUKS — *Linux Unified Key Setup*), tipicamente com AES; protege o disco **com a máquina desligada**. **Código aberto** (GPLv2).

- **BusKill** — cabo magnético "dead man's switch": ao ser desconectado (por exemplo, quando arrancam o notebook), dispara bloqueio ou desligamento. Hardware/software **código aberto** (GPLv3).
- **usbkill** — script que monitora as portas USB e desliga/apaga a máquina quando um dispositivo é inserido ou removido (dead man's switch por USB). **Código aberto** (GPLv3).

Referências

- HALDERMAN, J. Alex; SCHOEN, Seth D.; HENINGER, Nadia; et al. **Lest We Remember: Cold Boot Attacks on Encryption Keys**. USENIX Security Symposium, 2008.
- ULBRICHT, Ross William. Caso *United States v. Ross William Ulbricht* (Silk Road) — operador preso em 01/10/2013, na Biblioteca Pública de São Francisco, com o notebook **ligado e destravado**. Distrito Sul de Nova York, 2013–2015.
- BROŽ, Milan (mantenedor). **cryptsetup / LUKS On-Disk Format Specification**. Documentação do projeto cryptsetup.
- RUTKOWSKA, Joanna. **Evil Maid goes after TrueCrypt!** The Invisible Things Lab, 2009 (formulação do *evil maid attack*).
- BUSKILL PROJECT. **BusKill: a laptop kill cord dead man's switch**. Disponível em: https://www.buskill.in/

A Vigilância Global: 5, 9 e 14 Olhos

Existe uma frase que resume o espírito deste capítulo: tudo que é BOM pode ser usado para o MAL. O que antes não passava de uma conspiração sustentada por autores desacreditados, hoje é uma confirmação — e o leitor deve ler em paralelo o livro **THE SECRET HISTORY OF THE FIVE EYES**. No final da Segunda Guerra estava certo que haveria uma terceira, e que os indícios disso estariam nas informações. Se olhar com atenção, a Segunda Guerra poderia ter sido evitada, mas não foi: Hitler teve tempo, Hitler provou a fraqueza de seus adversários antes mesmo de pôr em prática sua máquina nefasta do mal. Foi dessa lição amarga que nasceu uma aliança para obter informações e usá-las para entender se há ou não perigo em um futuro próximo.

Como uma boa ideia vira uma quadrilha

Essa aliança iniciada por US, UK, Austrália, Canadá e Nova Zelândia tinha um viés bom, um viés positivo para a humanidade. Mas saiba: tudo que é BOM pode ser usado para o MAL. E logo essa organização focada em defesa iniciou um longo processo de quebra de anonimato e perseguição a indivíduos que, infelizes com os rumos de seus países, passaram a criticar governantes e as políticas públicas. Sempre uma boa ideia é desvirtuada por governantes. Hoje, segundo Edward Snowden — com provas —, esta instituição se tornou o maior esquema de vigilância digital e quebra de privacidade do planeta, e ainda por cima nunca foi alcançada pela **GDPR**, pela **LGPD** ou por qualquer outra lei de proteção de dados.

A base formal dessa aliança é um documento pouco conhecido do grande público: o **UKUSA Agreement**, um acordo secreto de partilha de inteligência de sinais assinado em 1946 entre Estados Unidos e Reino Unido, ao qual depois aderiram Canadá, Austrália e Nova Zelândia. Esses cinco países de língua inglesa formam o núcleo que ficou conhecido como os **Cinco Olhos (Five Eyes)**. O acordo permaneceu tão secreto que, por décadas, primeiros-ministros de países-membros sequer sabiam de sua existência; ele só foi oficialmente reconhecido e teve seu texto liberado em 2010. É este o alicerce jurídico da grande quadrilha que descrevo aqui — não uma invenção de teóricos da conspiração, mas um tratado com nome, data e assinatura.

ECHELON e a interceptação global

Muito antes de a internet existir na forma atual, essa aliança já operava uma máquina de escuta planetária. O nome que vazou para a imprensa foi **ECHELON**: uma rede de estações terrestres e antenas espalhadas pelo mundo, capaz de interceptar comunicações via satélite, chamadas telefônicas, faxes e, mais tarde, tráfego de dados. A lógica do ECHELON já antecipava o que veríamos décadas depois — não escutar um alvo específico com um mandado, mas aspirar tudo que passa pelo ar e filtrar depois por palavras-chave. Em 2001, o próprio Parlamento Europeu produziu um relatório reconhecendo a existência do sistema e alertando os cidadãos europeus de que suas comunicações estavam sendo capturadas por uma aliança estrangeira. Guarde a data: já em 2001 havia aviso oficial. O que faltava eram as provas internas — e elas viriam.

As provas de Snowden: PRISM e XKeyscore

Em 2013, um contratado da NSA chamado **Edward Snowden** entregou a jornalistas do *The Guardian* e do *Der Spiegel* dezenas de milhares de documentos internos. Foi o momento em

que a conspiração virou confirmação documentada. Dois programas ficaram famosos. O **PRISM** revelou que a NSA obtinha acesso direto a dados de usuários das maiores empresas de tecnologia americanas — e-mails, buscas, vídeos, arquivos armazenados — de forma sistemática. O **XKeyscore** era ainda mais assustador: descrito pela própria NSA como seu sistema de maior alcance, permitia a um analista, digitando apenas um endereço de e-mail ou um IP, vasculhar retroativamente quase tudo o que uma pessoa fez na internet — e-mails lidos, sites visitados, buscas feitas — muitas vezes sem qualquer autorização judicial prévia. Snowden resumiu o poder dessa ferramenta numa frase que ficou célebre: sentado à sua mesa, ele poderia grampear qualquer um, de você a um juiz federal, se tivesse o e-mail. Não é retórica: são os próprios slides internos da agência que provam a capacidade.

Cinco, nove e quatorze olhos

Este grande esquema, hoje essa grande quadrilha, não parou nos cinco fundadores. Ela se estende por Denmark (Dinamarca), France (França), Netherlands (Holanda), Norway (Noruega), Belgium (Bélgica), Germany (Alemanha), Italy (Itália), Spain (Espanha) e Sweden (Suécia), totalizando **14 participantes**. A expansão aconteceu em anéis concêntricos, cada anel com um grau de confiança e de partilha de dados menor que o de dentro.

Figura: Os anéis de 5, 9 e 14 olhos.

O anel mais interno são os **Cinco Olhos (Five Eyes)**: Estados Unidos, Reino Unido, Canadá, Austrália e Nova Zelândia — os signatários do UKUSA, que compartilham inteligência quase sem restrições entre si. O segundo anel forma os **Nove Olhos (Nine Eyes)**, que adicionam quatro países: Dinamarca, França, Holanda e Noruega. O terceiro anel forma os **Quatorze Olhos (Fourteen Eyes)** — cujo nome técnico é **SIGINT Seniors Europe (SSEUR)** —, acrescentando outros cinco: Alemanha, Bélgica, Itália, Espanha e Suécia. À medida que se afasta do centro, a confiança diminui, mas a rede de troca de sinais continua a mesma. Para o hacker, o que importa é a soma: são quatorze países cujos serviços de inteligência colaboram de forma coordenada e cujo território deve ser tratado como hostil ao anonimato.

O truque de espionar o vizinho e trocar

Aqui está a engenhosidade perversa do arranjo, e é preciso entendê-la para saber por que ele é tão difícil de combater na justiça. Quase todos esses países têm leis internas — muitas vezes constitucionais — que proíbem ou limitam severamente a espionagem dos próprios cidadãos

sem mandado. A aliança contorna essas proteções com um truque simples: cada país espiona os cidadãos do outro e depois troca os dados. A NSA americana não pode grampear livremente um americano, mas a inteligência britânica pode — e então repassa o resultado a Washington. O britânico recebe de volta o que a NSA coletou sobre britânicos. O dado que a lei interna proíbe coletar diretamente chega pela porta dos fundos, já pronto, vindo de um "parceiro". É a lavagem de vigilância: contornar a Constituição terceirizando a violação para um aliado. Por isso a quadrilha nunca foi alcançada pela GDPR nem pela LGPD — ela foi desenhada, desde o UKUSA de 1946, para operar exatamente nas frestas entre as jurisdições.

Por que isso destrói o anonimato no Tor

Toda pessoa que, dentro destes países, acessa algo ou faz algo que alguma dessas máfias não goste será perseguida no mundo digital em todas essas máfias — porque os dados fluem livremente entre elas. E o problema atinge diretamente as ferramentas de anonimato. O **Tor** tem uma fragilidade grave: mais de 90% dos nós de saída estão nestes países, e isso é péssimo. Como expliquei no capítulo do roteamento Onion, quem controla o nó de entrada **e** o nó de saída ao mesmo tempo, com alguma perícia, pode deduzir alguma coisa sobre o usuário por meio de um ataque de confirmação de tráfego. Se catorze países aliados operam ou observam a maioria desses nós e trocam entre si o que veem, o adversário "global" que a rede Tor sempre temeu deixa de ser hipótese e vira arranjo institucional. Não é que o Tor esteja quebrado — é que a geografia dos seus nós joga contra você.

A mesma regra vale para **VPNs**. Escolher um servidor VPN sediado nos Estados Unidos, no Reino Unido ou em qualquer um dos quatorze é entregar seu ponto de saída a uma jurisdição que pode ser legalmente obrigada a registrar e compartilhar. A regra prática que adoto e recomendo é direta: escolha servidores VPN e nós Tor **fora dos quatorze olhos** sempre que possível. Não é paranoia — é reduzir a superfície de ataque, tirar seus pontos de entrada e saída do território onde a partilha de dados é rotina.

Configurando o Tor para evitar os catorze olhos

Pode-se configurar o Tor para evitar os catorze olhos, e isso torna a conexão mais segura. Para que a rede não use tais países, deve-se editar o arquivo `/etc/tor/torrc` e acrescentar, no final do arquivo, as seguintes linhas:

```
  1 RunAsDaemon 1
256 ExcludeNodes {us},{gb},{ca},{au},{nz},{dk},{fr},{nl},{no},{be},{de},{it},{es},{se}
257 ExcludeExitNodes {us},{gb},{ca},{au},{nz},{dk},{fr},{nl},{no},{be},{de},{it},{es},{se}
258 StrictNodes 1
```

A linha `RunAsDaemon` habilita a execução do Tor na inicialização do GNU/Linux. Já as linhas `ExcludeNodes` e `ExcludeExitNodes` garantem que nenhum nó localizado nesses países será usado no encadeamento de nós da rede Tor — a primeira exclui os nós em qualquer posição do circuito, a segunda reforça especificamente o nó de saída, que é o mais sensível. Acrescentei ainda o `StrictNodes 1`, que torna a exclusão **obrigatória**: sem ele, o Tor pode ignorar a lista se não encontrar uma rota alternativa, e você acabaria voltando a passar justamente pelos países que queria evitar. Os códigos entre chaves são os códigos de país ISO de dois caracteres (`{us}` para Estados Unidos, `{gb}` para Reino Unido, e assim por diante). Mantenha essa lista sempre atualizada, porque a geopolítica muda e novos países podem aderir aos anéis.

Uma observação honesta, que vale mais do que a configuração em si: ao excluir tantos países, a navegação ficará muito lenta, porque você está descartando a maior parte dos nós disponíveis na rede. E corre-se o risco de uma URL `.onion` retornar 404, já que o circuito para alcançá-la pode não conseguir se montar dentro das restrições. É o preço do anonimato — mais segurança em troca de menos velocidade e menos alcance. Cada um decide, para cada tarefa, onde fica o ponto de equilíbrio.

Ferramentas citadas

- **Tor** — rede de roteamento anônimo (roteamento Onion); permite excluir países dos circuitos via `torrc`; **código aberto** (licença BSD de 3 cláusulas).

Referências

- KERBAJ, Richard. **The Secret History of the Five Eyes**: The Untold Story of the International Spy Network. London: John Blake Publishing, 2022.
- GREENWALD, Glenn; MacASKILL, Ewen. **NSA Prism program taps in to user data of Apple, Google and others**. The Guardian, 7 jun. 2013.
- GREENWALD, Glenn. **XKeyscore: NSA tool collects 'nearly everything a user does on the internet'**. The Guardian, 31 jul. 2013.
- POITRAS, Laura et al. **How the NSA Targets Germany and Europe**. Der Spiegel, 1 jul. 2013.
- PARLAMENTO EUROPEU. **Report on the existence of a global system for the interception of private and commercial communications (ECHELON interception system)**. Bruxelas, 2001.
- UKUSA AGREEMENT. Acordo de partilha de inteligência de sinais entre Reino Unido e Estados Unidos, 1946 (desclassificado e publicado em 2010).

Ativando um Celular e SMS Anônimo

Existe uma hora em que o anonimato de rede não basta e o mundo cobra um número de telefone: um cadastro que exige SMS, um serviço que só libera a conta depois de mandar um código, uma operação que precisa de uma voz do outro lado. Ativar um celular sem que ele se torne uma coleira é um dos processos mais delicados deste livro — não por ser tecnicamente difícil, mas porque a operadora, o aparelho e o seu próprio hábito de deslocamento trabalham juntos contra você. Este capítulo trata de como adquirir e ativar um celular deixando o mínimo de rastro, de por que o problema é mais o aparelho do que o chip, e de quando um serviço de SMS anônimo na internet resolve — ou custa dinheiro à toa.

O aparelho antes do chip

Como ativar um celular? Esse é um processo muito complexo e há muita discussão, e algumas observações minhas são controvérsias. Primeiro é o celular que se deve adquirir: deve-se utilizar um que **não possua um sistema operacional moderno**, isso pois a operadora faz um *fingerprint* do aparelho e de dados pessoais do indivíduo. Um smartphone comum — Android ou iOS — é, para efeito prático, um segundo delator dentro do seu bolso: ele carrega contas Google ou Apple, listas de redes Wi-Fi já conhecidas, um histórico de sensores e uma identidade de software que a operadora e os aplicativos correlacionam com tudo o que você já foi. Um aparelho básico, sem sistema operacional moderno (aqueles telefones de teclado, os *feature phones*), não tem essa mochila de metadados para entregar.

Mas o ponto que quase ninguém explica é que existem **dois** identificadores em jogo, e confundi-los é o erro clássico. O aparelho tem o **IMEI** (*International Mobile Equipment Identity*), um número gravado no próprio hardware que identifica o telefone físico. O chip tem o **IMSI** (*International Mobile Subscriber Identity*), gravado no cartão SIM, que identifica a assinatura — a "pessoa" perante a operadora. A rede registra o par: qual IMSI passou por qual IMEI. É por isso que trocar só uma das metades não adianta nada. Se você mantém o mesmo aparelho e só troca o chip, a operadora vê o IMEI velho aparecer com um IMSI novo e amarra os dois cadastros na mesma pessoa — o telefone "conta" que é o mesmo dono. Se você mantém o mesmo chip e troca de aparelho, o IMSI velho denuncia o IMEI novo pela mesma lógica. Anonimato de celular é sempre um par novo: **aparelho novo com chip novo**, e nunca cruzar esse par com nenhum outro que já esteja associado a você.

O ponto e o momento da ativação

O segundo ponto interessante é o ponto e o momento da ativação. O celular que foi adquirido **sem cartão de crédito** (no dinheiro) e **pessoalmente** em um local movimentado — tipo Santa Efigênia — deve ser ativado em uma **rota comum**, pois em caso processual com a máfia local há o álibi de ser um local que se passa para ir trabalhar em seu dia normal. Essa lógica do álibi é o coração do capítulo: se você ativa um celular em um local que nunca frequentou, não há álibi — a ativação vira um ponto isolado no mapa, fácil de casar com a sua identidade justamente por ser anômala. Ativando na rota que você percorre todos os dias, aquele registro se dilui em centenas de outros registros seus no mesmo lugar, e deixa de ser uma prova de que aquele número é seu.

Comprar em dinheiro e pessoalmente elimina a trilha financeira (não há fatura de cartão ligando você ao aparelho) e a compra online (que exige endereço de entrega e, quase sempre, um pagamento identificado). O local movimentado importa por dois motivos: você é mais um

rosto na multidão, e a própria compra tem cobertura — todo mundo compra eletrônico em Santa Efigênia, então estar ali não prova nada.

Outro ponto interessante é a **agilidade** que se deve ter para validar códigos de ativação. Hoje inúmeros sites pedem para cadastrar celular, e o código de SMS costuma ter validade curta. Você quer receber o número, disparar todos os cadastros que precisa validar e concluir isso rápido, num intervalo controlado, e não deixar o aparelho ligado dia após dia acumulando localização. Depois disso, **evita-se utilizar o celular em outros locais que são de seu cotidiano**, para evitar busca por restrição. Cada lugar do seu dia a dia em que você liga esse aparelho é mais um ponto que amarra o número anônimo à sua rotina real — o oposto do que o álibi tentou construir.

Por que ligar o aparelho já é entregar-se

A recomendação de não usar o celular nos seus lugares de sempre não é superstição: é consequência direta de como a rede celular localiza um aparelho. Para receber ligações e SMS, o telefone precisa estar registrado numa **célula** — a área de cobertura de uma antena (a *Estação Rádio-Base*). A operadora sabe, a cada momento, em qual célula cada IMEI/IMSI está registrado, e guarda isso nos **CDR** (*Call Detail Records*), os registros de chamada. Com a intensidade do sinal em três ou mais antenas, dá para fazer **triangulação** e estreitar a posição a um raio de dezenas ou centenas de metros — e num centro urbano denso, com muitas antenas, essa precisão só aumenta. Ou seja: o simples fato de o aparelho estar ligado já desenha, no histórico da operadora, um mapa de onde ele esteve e quando.

Some a isso um ataque mais ativo: o **IMSI catcher**, popularmente conhecido pela marca **Stingray**. É uma falsa antena — um aparelho que se passa por Estação Rádio-Base legítima e, por emitir o sinal mais forte da região, faz todos os celulares de uma área se conectarem a ele. Com isso, quem opera o IMSI catcher coleta os IMSI (e muitas vezes os IMEI) de **todo mundo** que estava naquele lugar naquele momento, e pode até forçar os aparelhos a downgrade para protocolos antigos e interceptar tráfego. Numa manifestação, num ponto de encontro, num bairro sob vigilância, o IMSI catcher transforma "estar presente com o celular ligado" em "estar na lista". É mais uma razão para o aparelho anônimo aparecer no menor número possível de lugares — e nunca nos lugares que já são seus.

O conceito de burner e a colocalização

O celular anônimo, de uso limitado e descartável, é o que o jargão chama de **burner phone** — o "telefone queimável", que se usa para uma finalidade e depois se destrói. E aqui vem a regra que anula todo o resto se for quebrada: **nunca ligue o burner perto do seu telefone real.** A rede registra, minuto a minuto, quais aparelhos estavam na mesma célula ao mesmo tempo. Se o seu telefone pessoal (o que tem seu nome, seu CPF, seu banco) e o burner aparecem repetidamente juntos — mesma antena, mesmos horários, mesmo trajeto —, um analista cruza os dois históricos e conclui, com altíssima confiança, que pertencem à mesma pessoa. É a **correlação de colocalização** (*co-location*): você não precisou ligar de um para o outro, não precisou trocar mensagem entre eles; bastou andarem juntos. O erro mais comum é levar o burner no mesmo bolso do celular de sempre "só para testar". Nesse instante, o anonimato acabou.

Na prática, isso significa: o burner fica desligado e longe do seu aparelho real; quando você vai operá-lo, o telefone pessoal fica em casa (ligado, na sua rotina normal, construindo o seu álibi de que você estava em outro lugar); e o burner só é ligado na rota e no momento planejados. Aparelho e chip são consumíveis: terminada a campanha, **ambos devem ser**

"queimados" — o chip destruído e o aparelho descartado ou destruído, nunca reaproveitado num próximo ciclo, sob pena de o IMEI antigo reamarrar tudo.

Os dados que a operadora pede

Sobre os dados, tal como nome e CPF — já sabe, a internet é foda. No Brasil a habilitação de chip é atrelada ao CPF, e é aí que mora boa parte da dificuldade. **Cuidado que há operadoras que pedem para envio de fotos ao final** — selfie, foto do documento, prova de vida —, e uma foto sua amarrada ao chip destrói qualquer anonimato de uma vez. **Escolha bem a operadora**: as exigências variam, e o que se quer é o caminho de habilitação que peça o mínimo e não capture imagem sua. Vale entender que essa foto não é implicância da empresa; é uma trilha que fica guardada e pode ser recuperada muito depois do fato, junto com o CDR e o par IMEI/IMSI.

SMS anônimo online: quando serve e quando é cilada

Caso não queira um celular, na internet existem inúmeros serviços de **SMS anônimo** — vários —, mas o autor deste livro já perdeu uns trocados com alguns. Procure antes para ter certeza da idoneidade do serviço, pois tem que validar:

- **Se realmente entregam o serviço** — muitos cobram e simplesmente não recebem o código, ou o número já vem bloqueado pela plataforma que você quer cadastrar.
- **Se o número é usado por uma grande quantidade de pessoas** — números públicos e compartilhados são reciclados entre milhares de usuários, então o serviço-alvo (Google, Telegram, etc.) já os conhece e recusa, e pior: quem teve o número antes pode ter deixado contas amarradas a ele, e quem vier depois pode recuperar as suas.

Há um ponto que precisa ficar claro sobre depender de SMS para autenticar: **SMS como segundo fator (2FA) é fraco**, como já foi visto no capítulo de Identidade. O SMS trafega por uma sinalização telefônica antiga, o **SS7**, cheia de brechas que permitem interceptar mensagens de terceiros; e existe o **SIM swap**, em que o atacante convence a operadora a portar o seu número para um chip dele e passa a receber os seus códigos. Ou seja: usar um número — próprio ou alugado — como chave de segurança é apoiar-se na parte mais frágil da corrente. Quando puder, prefira aplicativos autenticadores ou chaves físicas; o número de telefone é para satisfazer o cadastro que exige um, não para proteger a conta.

E o cerco aperta: atualmente **até o famoso Protonmail está pedindo número telefônico** em muitos cadastros, lastimável — justamente um serviço que se vende como privado passou a condicionar a criação de conta a um telefone (em geral quando detecta acesso via Tor ou VPN, para conter abuso). É mais um lembrete de que o número virou um gargalo de identidade que os serviços usam para atar você a algo do mundo físico — e por isso mesmo vale tanto a pena tratá-lo com o cuidado deste capítulo.

Ferramentas citadas

- **Serviços de SMS anônimo / números descartáveis** (categoria; ex.: sites de *disposable numbers*) — alugam números temporários para receber códigos; **comerciais**, idoneidade variável — validar entrega e se o número é compartilhado antes de pagar.
- **Protonmail** — serviço de e-mail com criptografia ponta a ponta; **freemium** (núcleo de código aberto), passou a exigir verificação por telefone em parte dos cadastros.

Referências

- BRASIL. **Lei Geral de Proteção de Dados Pessoais (LGPD)**, Lei nº 13.709/2018 — tratamento de dados pessoais como CPF e imagem em cadastros.
- ENGEL, Tobias. **SS7: Locate. Track. Manipulate.** Chaos Communication Congress (31C3), 2014.
- FEDERAL COMMUNICATIONS COMMISSION (FCC). **Protecting Consumers from SIM Swap and Port-Out Fraud**. Report and Order, 2023.
- AMERICAN CIVIL LIBERTIES UNION (ACLU). **Stingray Tracking Devices: Who's Got Them?** Disponível em: https://www.aclu.org/
- 3GPP. **TS 23.003 — Numbering, addressing and identification** (definição de IMEI e IMSI).

Comunicação por XMPP

Como foi o fim do LulzSec? Basicamente os integrantes do grupo, que possuíam várias contas em chans, aos poucos se entregaram; na verdade o elo fraco era Hector Xavier Monsegur, que além de não saber esconder sua vida pessoal ainda conectou apenas 1 vez em um chat sem proteção do Tor. Que falta fez o Kodachi. Um deslize, uma única conexão exposta, e a linha inteira se desenrola — vale para o grupo mais barulhento da história do hacktivismo como vale para você.

Antes de entrar neste mundo, o hacker deve aprender a criar enredo, narrativas falsas e se manter nas narrativas. O Deus Loki deve ser fichinha perto de um bom hacker, tudo isso para despistar curiosos e manter sua segurança. Além das narrativas, temos as ferramentas e os meios; nesta seção vamos falar sobre as ferramentas e os meios, visto que a narrativa foi descrita em capítulos anteriores. Mas saiba, sobre ferramentas: existem enormes listas de discussões sobre pontos fortes e pontos fracos, e saiba que nada será perfeito.

O que é o XMPP

Hackers precisam se comunicar — e o meu XMPP, para registro, é `nao.importa.web@xmpp.jp`. O e-mail não é uma ferramenta tão interessante, principalmente com as recentes notícias sobre o ProtonMail (ver 2021); outra forma interessante são as ferramentas de mensagens instantâneas (**instant messaging**, IM). Esses aplicativos fornecem não apenas os meios para os usuários se comunicarem com outras pessoas em tempo real, mas também obter suas informações de presença.

Um dos primeiros protocolos de IM abertos foi o **Jabber**, que começou como um protocolo de IM não padronizado em 1998. Como um protocolo extensível construído com XML, o Jabber rapidamente encontrou outros usos como um transporte geral ou middleware orientado a mensagens (**message-oriented middleware**, MoM). Eventualmente, o XMPP surgiu do Jabber como um protocolo baseado em padrões, na forma de um documento de grupo de trabalho da IETF: a **RFC 3920**, "Extensible Messaging and Presence Protocol (XMPP)" — hoje substituída pelas RFC 6120 e RFC 6121, como veremos nas referências.

O nome **XMPP** (**Extensible Messaging and Presence Protocol**) carrega, letra a letra, o que o protocolo faz:

- **Protocolo** — o XMPP é um conjunto de padrões que permite que sistemas se comuniquem entre si. É amplamente usado na web, mas geralmente não é anunciado;
- **Presença** (**Presence**) — o indicador de presença informa aos servidores que você está online, offline ou ocupado. Em termos técnicos, a presença determina o estado de uma entidade XMPP;
- **Mensagens** (**Messaging**) — é a parte em que você vê a mensagem instantânea (IM) trafegando entre clientes. O XMPP foi projetado para enviar todas as mensagens em tempo real usando um mecanismo *push* muito eficiente;
- **Extensível** (**Extensible**) — definido em um padrão aberto e usando uma abordagem de sistemas abertos de desenvolvimento e aplicação, o XMPP foi projetado para ser extensível. Em outras palavras, ele foi projetado para crescer e acomodar mudanças.

A **XMPP Standards Foundation** (também conhecida como **XSF** e anteriormente Jabber Software Foundation) é uma organização de desenvolvimento de padrões independente e sem fins lucrativos, cuja missão principal é definir protocolos abertos para presença, mensagens

instantâneas e comunicação e colaboração em tempo real no topo do XMPP. O XSF também fornece informação e infraestrutura para a comunidade mundial de desenvolvedores Jabber/XMPP, provedores de serviços e usuários finais. Além disso, o XSF administra o Programa de Licenciamento de Marcas Registradas Jabber.

Uma rede descentralizada de servidores

O que torna o XMPP diferente de um WhatsApp da vida é a arquitetura. Existe uma extensa rede de servidores descentralizados pelo mundo, e o usuário deve estar atento às leis do país bem como à integridade do grupo que controla o servidor escolhido. Cada servidor é um domínio independente que conversa com os outros — assim como e-mails de provedores diferentes trocam mensagens entre si, um usuário em `jabber.de` fala com um usuário em `xmpp.jp` sem que exista uma empresa central no meio. Isso é a **federação**: não há um dono do protocolo, e derrubar um servidor não derruba a rede.

Quem sabe a Alemanha seja um lugar mais discreto? Mas pertence aos 14 olhos, e isso é muito estranho. É o tipo de ponderação geopolítica que você deve fazer antes de escolher onde abrir uma conta: reputação do operador, jurisdição do servidor e a companhia que essa jurisdição mantém nas alianças de vigilância.

As três regras da comunicação segura via XMPP

O XMPP é, de longe, a melhor solução nos casos em que é necessário o nível máximo de proteção de comunicação com mensagens instantâneas. Por padrão, o protocolo XMPP não possui criptografia confiável de ponta a ponta, mas você pode adicioná-la. Para tornar a comunicação via XMPP o mais anônima e segura possível, execute as seguintes etapas:

- a conexão com o servidor Jabber/XMPP deve ser feita apenas através do Tor;
- sempre use criptografia OTR/PGP (e, hoje, prefira **OMEMO** — veremos adiante);
- use login aleatório, como `alsyyyqfhfeyqfquf@jabber.com`.

Lembre-se dessas três regras fundamentais de comunicação segura via XMPP; outra recomendação importante é utilizar o Tor. Existem vários clientes XMPP; abaixo temos uma lista de possíveis ferramentas que você pode utilizar, organizadas por sistema operacional.

| Cliente XMPP | Sistema operacional |

| --- | --- |

| Aparté | BSD / Linux / macOS |

| AstraChat | Android / iOS / Linux / macOS / Windows |

| BeagleIM (by Tigase, Inc.) | macOS |

| blabber.im | Android |

| Bruno the Jabber™ Bear | Android |

| Conversations | Android |

| Gajim | Linux / Windows |

| Kaidan | Android / Linux / macOS / Windows |

| Monal IM | iOS / macOS |

| Poezio | Linux / macOS |

| Psi e Psi-plus | Linux / macOS / Windows |

| Pàdé | Browser |

| SiskinIM (by Tigase, Inc.) | iOS |

| Spark | Linux / macOS / Windows |

| StorkIM (by Tigase, Inc.) | Android |

| Swift | Linux / macOS / Windows |

| UWPX | Windows |

| yaxim | Android |

Cliente PSI-Plus

Neste exemplo será utilizado o Psi-plus, mas pode ser utilizada qualquer ferramenta da tabela acima; já no próximo tópico vou demonstrar o Gajim. Para iniciar, vamos assegurar que tanto o Psi-plus está instalado quanto o OpenPGP; para isso, digite no terminal os seguintes comandos.

```
  1 sudo apt install psi-plus -y
259 sudo apt install gnupg -y
260 sudo apt install libsasl2-modules -y
```

Também é recomendado que se instale o Tor para aumentar o grau de anonimato da sua troca de mensagens. Saiba como instalar o Tor pelo capítulo dedicado a ele neste livro. Após a instalação, inicie o programa Psi-plus com o comando abaixo.

```
  1 psi-plus
```

Pronto. Ao iniciar a aplicação, caso não tenha nenhuma conta XMPP, clique em "Register new account"; neste material será criada uma nova conta. Primeiro ponto interessante para se discutir: existem inúmeros servidores XMPP, mas você deve escolher um servidor XMPP que:

- tenha reputação;
- permita criação de usuários pelo próprio cliente XMPP.

A reputação é importante. Existem inúmeros entusiastas que sobem servidores XMPP e não possuem o conhecimento técnico para blindar o serviço; o próprio serviço XMPP vira alvo de ataques que visam à invasão. Já o fato de o servidor permitir que o cliente XMPP faça o cadastro de novas contas é importante para quem não quer abrir um Browser e se expor. No exemplo vou escolher o serviço `jabber.de`, por se tratar de um servidor conhecido.

Ainda nesta interface, clique em "Edit..." na opção Proxy; vamos configurar a conexão por Tor e já de cara aumentar o anonimato. Clique em "New" e, logo em seguida, escolha "SOCKS 5"; informe o IP da máquina local e a porta do serviço Tor, que no GNU/Linux é a **9050**. Clique em "Save" e a interface será fechada. Veja que agora a configuração Tor criada aparece como opção selecionada para Proxy.

Avance pressionando "Next"; agora está na hora de criar um login "nome" e uma senha. Utilize uma senha forte e, principalmente, "uma senha que não usa em lugar nenhum" — não ligue essa conta a você por uma senha. Caso use esta conta para algo muito, muito sigiloso, não utilize como login um nome de algo ligado a você; de preferência não use nenhuma palavra encontrada em dicionário (é exatamente o espírito daquele login `alsyyyqfhfeyqfquf@jabber.com`). Clique em avançar.

Alguns servidores pedem e-mail, telefone etc.; é natural que uma boa escolha sejam servidores como o `jabber.de`, que não é rigoroso no cadastro. Agora chegou a hora de testar sua conta. Vou simular a conversa entre 2 contas usadas para testes e desenvolvimento deste conteúdo, mas antes vamos confirmar — sempre — se a configuração do proxy está correta. Clique em "Modify Account..." e, na aba "Connection", confirme que o proxy está selecionado conforme configurado no processo de criação da conta.

Agora ative o status "Online" para conversar com os demais usuários; neste exemplo vou utilizar outra conta de testes. Após estar online, clique sobre a conta e "Inicie uma nova conversa". Troque mensagens com o outro usuário: sucesso.

Cliente Gajim

Um cliente XMPP que está ganhando muitos adeptos é o **Gajim** — realmente vem me surpreendendo, e venho usando este aplicativo em algumas VMs em que atuo como hacker. A instalação é muito simples.

```
  1 sudo apt install gajim -y
261 sudo apt install python3-gnupg -y
262 sudo apt install gajim-openpgp -y
```

No primeiro comando o Gajim é instalado; mas, para se manter a segurança, é importante instalar o `python3-gnupg` (o Gajim utiliza Python) e o `gajim-openpgp` para adicionar a criptografia de ponta a ponta. Sem o PGP, toda comunicação é criptografada só entre os clientes e o servidor, mas não há a segunda criptografia, que é a criptografia entre os clientes.

Sempre, sempre utilize o Tor; para isso, instale o Tor — e é natural que neste material você encontre tal operação. Com o status válido de que o Tor está rodando, inicie o Gajim, seja por linha de comando, seja pelo menu do seu GNU/Linux gráfico. Ao abrir, terá que adicionar a primeira conta; para isso, informe o usuário + domínio XMPP. Coloque também a senha (lógico) e marque as opções avançadas. Clique em "Log In". Selecione a conexão por proxy Tor e coloque o domínio XMPP em que criou sua conta (o mesmo usuário + servidor XMPP que digitou na tela anterior).

Faça o login e pronto: abrirá uma lista vazia a princípio, pois você precisa de amiguinhos. Então é só colar na galera; pode entrar em contato com a staff hacker pela sala:

```
1 xmpp:staff_hacker@conference.xmpp.jp?join
```

Quando entrar em contato com um outro usuário XMPP, combine com ele ativar o PGP — é muito simples. Com o chat aberto com o usuário, clique no cadeado e escolha a opção "OpenPGP". Se a mensagem de verificação aparecer, a outra parte também tem que fazer essa operação PGP. Depois que ambos fizerem, repare que as mensagens aparecem com um cadeado verde, indicando que aquela mensagem está criptografada.

E aqui vai o aviso que nunca é demais repetir: **ATENÇÃO — em GRUPOS não há criptografia ponto a ponto**, pois é multiusuário. Então combine de falar no privado o que for mais sensível; para algumas conversas, o par a par é o único terreno seguro.

OMEMO: a criptografia moderna do XMPP

Até aqui segui o roteiro do OTR e do PGP porque é o que a maioria dos tutoriais ensina — e porque funciona. Mas seria desonesto de minha parte não te contar que, desde por volta de 2015, existe algo melhor, e que hoje é o padrão que eu recomendo de fato: o **OMEMO** (a sigla

é um acrônimo recursivo, "OMEMO Multi-End Message and Object Encryption"), padronizado pelo XSF como a extensão **XEP-0384**. Ele traz para o XMPP o mesmo mecanismo criptográfico que blinda o Signal — o **Signal Protocol**, com seu algoritmo **Double Ratchet** — só que dentro de uma rede federada e aberta, sem depender de nenhuma empresa.

A vantagem central do OMEMO tem nome: **sigilo futuro** (**forward secrecy**). As chaves de sessão giram a cada mensagem; se um adversário capturar a chave de hoje, ele não consegue decifrar as mensagens de ontem, nem as de amanhã. Some a isso o suporte nativo a **múltiplos dispositivos**: você pode ter o mesmo contato conversando do celular e do notebook, cada aparelho com sua própria chave, e a mensagem chega cifrada para todos eles. É por essas duas propriedades que hoje o OMEMO é preferível tanto ao OTR quanto ao PGP.

Vale entender por que os dois veteranos envelheceram. O **OTR** (**Off-the-Record Messaging**) foi revolucionário no seu tempo e até tem sigilo futuro, mas nasceu para um mundo síncrono e de **um único dispositivo**: as duas pontas precisam estar online ao mesmo tempo para negociar a sessão, e ele não sabe lidar com você respondendo do celular e do desktop. Numa era de mensageria que precisa funcionar com a outra pessoa offline e com aparelhos múltiplos, esse modelo simplesmente não acompanha.

O **PGP**, por outro lado — aquele mesmo que ativamos no cadeado do Gajim —, resolve o problema de estar offline, mas paga caro por isso: ele **não tem sigilo futuro**. É a mesma chave de longo prazo cifrando tudo; comprometeu a chave privada uma vez, o adversário decifra todo o histórico que tiver arquivado. Pior: o PGP protege o corpo da mensagem, mas deixa os **metadados expostos** — assunto, quem falou com quem e quando ficam legíveis. Ele continua útil para assinar e cifrar e-mails e arquivos, mas para bate-papo instantâneo é a escolha mais fraca das três.

Ativar o OMEMO na prática é tão simples quanto o PGP: em clientes modernos como o **Conversations** (Android) ou o próprio Gajim com o plugin correspondente, basta clicar no cadeado da conversa e escolher "OMEMO" em vez de "OpenPGP". O cliente troca as chaves automaticamente; a única tarefa manual — e importante — é **verificar as impressões digitais (fingerprints)** dos dispositivos do seu contato, de preferência por um canal fora do XMPP ou lendo o QR code pessoalmente, para não cair num ataque de intermediário (**man-in-the-middle**).

Mas guarde a limitação que nenhuma criptografia de conteúdo resolve: mesmo com OMEMO perfeito, o **metadado continua no servidor**. Quem falou com quem, a que horas, com que frequência, de qual endereço IP — isso o servidor XMPP registra, porque precisa dessa informação para entregar a mensagem. O OMEMO embaralha o *conteúdo*, não a *relação*. É por isso, e só por isso, que a primeira das três regras — conectar apenas pelo Tor — não é opcional: o Tor é o que esconde o IP e a relação de quem-fala-com-quem que o OMEMO, sozinho, deixa às claras.

Para situar o XMPP+OMEMO no mapa das mensageiras seguras, duas comparações rápidas e honestas. O **Signal** usa exatamente a mesma criptografia (aliás, o OMEMO nasceu do protocolo do Signal), mas é **centralizado** — roda nos servidores de uma única fundação nos Estados Unidos — e **exige um número de telefone** para cadastro, o que amarra a conta a uma identidade civil e, para o hacker, é um problema sério de opsec. Já o **Matrix** é, como o XMPP, uma rede **federada** de servidores independentes, com criptografia de ponta a ponta própria (baseada no algoritmo Olm/Megolm, primo do Double Ratchet); é uma alternativa legítima e mais moderna na experiência de uso, embora historicamente também vaze metadados entre os servidores da federação. Nenhuma dessas ferramentas é perfeita — como avisei lá no começo, nada será. O que muda é onde cada uma coloca a sua confiança: no

Signal, numa empresa; no XMPP e no Matrix, na sua própria escolha de servidor e na disciplina de sempre passar pelo Tor.

Ferramentas citadas

- **Psi-plus** — cliente XMPP para desktop (Linux/macOS/Windows), fork do Psi com recursos extras e suporte a OTR/PGP; **código aberto** (GPL).
- **Gajim** — cliente XMPP em Python para Linux/Windows, com plugins para OpenPGP e OMEMO; **código aberto** (GPL v3).
- **Conversations** — cliente XMPP para Android, referência em suporte a OMEMO; **código aberto** (GPL v3).
- **OMEMO** — extensão de criptografia de ponta a ponta para XMPP (XEP-0384), baseada no Signal Protocol, com sigilo futuro e múltiplos dispositivos; **especificação aberta** do XSF.
- **OpenPGP / GnuPG (gnupg)** — implementação livre do padrão OpenPGP, usada para cifrar e assinar mensagens e arquivos; **código aberto** (GPL v3).

Referências

- SAINT-ANDRE, Peter. **RFC 6120 — Extensible Messaging and Presence Protocol (XMPP): Core**. IETF, 2011. Disponível em: https://datatracker.ietf.org/doc/html/rfc6120. (Substitui a RFC 3920.)
- SAINT-ANDRE, Peter. **RFC 6121 — Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence**. IETF, 2011. Disponível em: https://datatracker.ietf.org/doc/html/rfc6121. (Substitui a RFC 3921.)
- XMPP STANDARDS FOUNDATION. **XEP-0384: OMEMO Encryption**. Disponível em: https://xmpp.org/extensions/xep-0384.html.
- XMPP STANDARDS FOUNDATION. **History of XMPP** e documentação da XSF. Disponível em: https://xmpp.org/about/.
- MARLINSPIKE, Moxie; PERRIN, Trevor. **The Double Ratchet Algorithm** e **The X3DH Key Agreement Protocol**. Signal, 2016.

Comunicação por E-mail

Um serviço de e-mail permite que uma ou mais pessoas se comuniquem de forma **off-line**: tanto quem envia quanto quem recebe contam com servidores de armazenamento temporário de mensagens, e essas mensagens podem estar em texto plano (conforme Tanenbaum) ou criptografadas. É essa característica — a mensagem fica **parada** num servidor esperando ser lida — que separa o e-mail de uma conversa em tempo real e que, como veremos, o torna estruturalmente ruim para quem busca anonimato. Neste capítulo eu descrevo como o e-mail funciona por baixo, por que hackers costumam abominá-lo, quando ainda assim vale a pena ter uma caixa, e um caso real de quebra de privacidade que ficou famoso: o da ProtonMail.

Os protocolos: SMTP, POP e IMAP

Os protocolos utilizados nesta troca de dados são o **SMTP (Simple Mail Transfer Protocol)** e o **POP/IMAP (Post Office Protocol / Internet Message Access Protocol)**. Eu dedico um tópico inteiro só para discutir o comportamento destes protocolos em rede, inclusive por serem sempre alvos de hackers na rede local. A divisão de tarefas é simples: o **SMTP** cuida do **envio** — é ele que empurra a mensagem do seu cliente para o seu servidor e do seu servidor para o servidor do destinatário. Já o **POP** e o **IMAP** cuidam da **leitura**, isto é, de trazer a mensagem que está parada no servidor até você. A diferença entre os dois é que o POP tende a **baixar e apagar** (a mensagem migra para o seu computador), enquanto o IMAP **mantém tudo no servidor** e apenas sincroniza — o que, do ponto de vista de retenção, significa que suas mensagens continuam guardadas lá fora por tempo indeterminado.

O serviço de e-mail é trivial de montar. Eu ensino a instalação completa do servidor de e-mail **Postfix** no curso de GNU/Linux, seguindo uma arquitetura de solução com o seu servidor de saída (SMTP), o servidor de armazenamento e os servidores do outro lado. Guarde essa imagem mental do caminho da mensagem, porque é justamente ao longo dela que aparecem os problemas.

Os cinco problemas de segurança

Neste cenário temos alguns problemas que levam à insegurança da privacidade da informação. São eles:

1. **Em trânsito, usuário/senha podem ser transmitidos** (ver o caso dos monges no caso Lazarus) — ou seja, as credenciais podem trafegar de forma que alguém no caminho consiga capturá-las.
2. **Se criptografado, falha de segurança por perda de compromisso da chave de criptografia** — se a chave vaza ou é comprometida, toda a proteção cai por terra.
3. **Falha de segurança nos servidores que armazenam a mensagem** — a mensagem fica parada num servidor de terceiro, e esse servidor pode ser invadido ou entregar seu conteúdo.
4. **LOG de troca de mensagens** — o registro de quem falou com quem, quando e de onde.
5. **Governo.**

Com a criptografia entre quem envia e quem recebe (ponta a ponta) resolve-se o problema 3 e o problema 2; e com a criptografia entre os elementos, cada conexão com uma criptografia própria, resolve-se o problema 1. Outra coisa de que precisamos também é um serviço **ZERO**

LOG, para resolver o problema 4. Mas o problema 5 é impossível de solucionar — e é exatamente por isso que muitos hackers abominam o uso de e-mails.

PGP/GPG: o que a criptografia de ponta a ponta resolve e o que não resolve

Quando eu digo "criptografia entre quem envia e quem recebe", na prática do e-mail isso costuma significar **PGP (Pretty Good Privacy)** ou sua implementação livre, o **GnuPG (GNU Privacy Guard)**. A ideia é a clássica criptografia assimétrica: você publica sua **chave pública**, qualquer um cifra uma mensagem para você com ela, e só a sua **chave privada** consegue abrir. Isso ataca de frente os problemas 2 e 3 da lista acima — mesmo que o servidor de armazenamento seja invadido, o corpo da mensagem sai de lá como um bloco cifrado ilegível. É uma ferramenta poderosa, e vale ter no cinto.

O que não se pode fazer é confundir "conteúdo cifrado" com "comunicação anônima". O PGP/GPG tem limitações estruturais que precisam ser ditas com todas as letras, porque é aqui que muita gente se ilude:

- **Ele cifra o corpo da mensagem, mas não cifra o assunto (Subject).** O campo de assunto viaja em texto plano — escreveu "Reunião sobre o protesto de sábado" no assunto, entregou o jogo.
- **Ele não esconde os metadados.** Os cabeçalhos **To**, **From**, **Date**, os servidores por onde a mensagem passou (`Received:`) e o horário continuam todos expostos. E, como se discute ao longo deste livro, é o metadado — quem falou com quem, quando e com que frequência — que mais interessa a quem vigia. A criptografia protege o **conteúdo**, não o **grafo social**.
- **Ele não tem forward secrecy (sigilo futuro).** Diferente de protocolos modernos de mensageria, que trocam de chave a cada sessão, no PGP a mesma chave privada abre tudo. Se essa chave vazar **um dia** — apreensão do computador, senha fraca, malware —, o adversário decifra **retroativamente** todo o histórico de mensagens que ele porventura tenha guardado. Isso é o problema 2 da lista, na sua forma mais cruel.
- **Ele depende de gestão de chaves feita por humanos**, e humanos erram: publicam a chave errada, perdem a senha (o problema 2 de novo), não verificam a impressão digital (fingerprint) e caem num ataque de chave falsa.

Ou seja: PGP/GPG é excelente para blindar o **conteúdo** de um e-mail que você faz questão de mandar por e-mail. Não é, e nunca foi, uma capa de invisibilidade.

Para que serve ter uma caixa de e-mail

Diante de tantos problemas, a pergunta honesta é: qual o motivo para se ter uma caixa de e-mail? Pode ser:

- Para conversar com pessoas comuns;
- Todo sistema pede um e-mail (menos o cybersecurity.online);
- Para se passar por um ser humano normal e interagir com algum mecanismo;
- Para ataques, principalmente de **phishing**.

Repare que nenhum desses motivos é "manter uma comunicação secreta e anônima de longo prazo". O e-mail é uma **credencial de identidade civil** na internet — a chave que abre cadastros — e uma ferramenta de operação. Para conversa sensível, existem tecnologias melhores, como veremos no fim do capítulo.

Como escolher um serviço de e-mail

Quando for procurar por um serviço, você deve:

- Validar se o serviço é **criptografado**;
- Verificar se ele utiliza protocolos com **SSL** ou **TLS**;
- Ter uma **interface WEB**, nunca usar o Outlook ou qualquer cliente SMTP/POP no seu computador;
- Ser **ZERO LOG**.

O ponto da interface WEB merece um parágrafo, porque contraria o hábito da maioria das pessoas. Um cliente local (Outlook, Thunderbird e afins) deixa rastros no **seu** computador: baixa as mensagens, guarda cache, mantém a conta configurada, salva anexos — e se a máquina for apreendida ou infectada, está tudo ali. Além disso, um cliente fala SMTP/POP/IMAP direto, muitas vezes por caminhos que você não controla. A interface WEB, acessada sempre por cima de um **proxy ou VPN** (recomendo enfaticamente: "sempre use proxy ou vpn para acessar serviços de e-mail"), mantém os dados do lado do servidor e não gruda uma identidade permanente na sua máquina.

O caso ProtonMail: um caso real de quebra de privacidade

No caso ProtonMail eu vou descrever um caso real de quebra de privacidade de informações em que, ilegal ou não, a prisão do alvo foi executada — e a lição que fica é grande. No meu ponto de vista (discordando de Tanenbaum aqui), a maior quebra de privacidade não foi técnica, foi a **decisão da ProtonMail de liberar dados de usuários**. Mas por que isso é ruim? A resposta é simples: o problema é que a ProtonMail sempre ofereceu um serviço dito seguro quanto à privacidade — afinal, este é o slogan da empresa. Inclusive, vamos entender uma coisa: a própria empresa **informava que não armazenava o IP dos clientes**, conforme registro na **Wayback Machine**. Já depois do envio de dados para a Europol, a empresa mudou o discurso (afinal, liberou o IP associado a uma conta). E aí fica a pergunta óbvia: se não guarda, como tinha para liberar?

O caso concreto, para não pairar no vago: em 2021, autoridades francesas investigavam um **ativista climático** ligado ao movimento **Youth for Climate** na França, que usava uma conta ProtonMail. Por meio da cooperação policial internacional — o pedido chegou à Suíça através do canal da **Europol** e de uma requisição da autoridade suíça —, a ProtonMail recebeu uma **ordem judicial válida na Suíça** e, cumprindo-a, passou a **registrar o endereço IP** de acesso daquela conta, entregando-o em seguida. Não foi um "vazamento" nem uma invasão: foi a empresa cumprindo uma ordem legal do país onde opera. É por isso que a defesa dela girou toda em torno da **jurisdição suíça**.

A defesa da Proton, ponto por ponto

Temos que lembrar que, no pano de fundo, houve uma guerra de discursos entre governos sobre política ambiental. Segundo **Andy Yen** (fundador e CEO da Proton), "também estamos profundamente preocupados com este caso e lamentamos que as ferramentas legais para crimes graves estejam sendo usadas dessa maneira". A Proton reforçou oito pontos em sua defesa:

1. Sob nenhuma circunstância a criptografia deles pode ser ignorada;
2. O Proton Mail não fornece dados para **governos estrangeiros**;

3. As autoridades suíças só aprovam solicitações que atendam aos **padrões legais suíços**;
4. A transparência com a comunidade de usuários é extremamente importante para eles;
5. De acordo com a lei suíça, é obrigatório que um usuário seja notificado se um terceiro solicitar seus dados privados e esses dados forem usados em um processo criminal;
6. De acordo com a lei suíça atual, e-mail e VPN são tratados de forma diferente, e o **Proton VPN não pode ser obrigado a registrar dados** do usuário;
7. Devido à estrita privacidade da Proton, eles não sabem a identidade dos usuários e em nenhum momento sabiam que os usuários visados eram ativistas climáticos;
8. Não havia possibilidade legal de resistir ou lutar contra esse pedido específico.

Este caso mostra que o Proton Mail funciona como foi projetado. A identidade e a localização do ativista **já eram conhecidas** pelas autoridades francesas; portanto, o que elas provavelmente buscavam era o **conteúdo** dos e-mails, que poderia fornecer mais provas incriminatórias. E o fato é que o Proton Mail **não conseguiu entregar nenhuma mensagem**, mesmo sob ordem legal — o que prova que a criptografia de conteúdo funciona e, muito provavelmente, seria de grande ajuda para o ativista no processo. Se ele estivesse usando qualquer outro provedor comum de e-mail, o resultado teria sido muito pior. A Proton fecha com: "Entendemos suas preocupações e estamos com você — também somos ativistas."

A lição do caso

Em defesa do ProtonMail, e sendo técnico e honesto: parece que, do ponto de vista da engenharia, nada mudou. Eles provavelmente **não mantinham nenhum log de IP por padrão**. O detalhe fatal é outro: com apenas uma solicitação rápida por meio de um tribunal suíço, um governo ou agência externa pode fazer com que a empresa **passe a registrar** rapidamente todos os logs necessários para encontrar e prender o usuário do ProtonMail que estão procurando. Não guardar por padrão não é o mesmo que não poder ser obrigado a guardar a partir de amanhã.

O aprendizado que se obtém com este caso é o seguinte: **não importa onde está o servidor, o governo sempre consegue o que quer** — o que se pode fazer é **dificultar** o trabalho desses agentes. A Suíça tem fama de privacidade, e ainda assim a jurisdição suíça foi suficiente para virar a chave. Minhas recomendações a partir daí:

1. Todos têm a liberdade de falar e de naturalmente protestar (sem violência) sobre o que quiserem;
2. Leia a matéria original no site Technadu, e o comunicado da própria Proton, para formar sua opinião;
3. Use e-mail só para se comunicar com pessoas normais sobre coisas normais.

E some a isso o que já vimos sobre PGP/GPG: mesmo que o conteúdo estivesse cifrado ponta a ponta (e no caso ProtonMail estava), o **IP de acesso** e os **metadados** não estão dentro dessa proteção. Foi exatamente por essa fresta — o registro do IP passou a ser feito sob ordem — que o cerco se fechou. A criptografia protegeu o conteúdo; a jurisdição pegou o resto.

Por que o e-mail é estruturalmente ruim para o anonimato

Vale consolidar, porque não é acidente nem má-fé de uma empresa específica — é a natureza do e-mail. O protocolo SMTP é **antigo**, nasceu numa internet acadêmica onde ninguém pensava em anonimato, e por isso os **metadados sempre viajam expostos**: remetente, destinatário, horários, servidores intermediários, tudo em texto plano nos cabeçalhos. A

entrega é **store-and-forward** (armazena e encaminha): a mensagem fica **retida** em servidores de terceiros — o seu e o do outro lado —, e retenção significa que existe uma cópia esperando para ser pedida por quem tiver poder para pedir. E, por ser federado, uma única mensagem cruza a jurisdição de **vários** países entre a origem e o destino, multiplicando o número de governos que podem, cada um sob suas leis, exigir os registros. Some tudo: protocolo velho, metadados sempre à mostra, dados retidos fora do seu controle e jurisdição espalhada. É a receita oposta à do anonimato. Por isso a frase se repete: hackers abominam e-mail — não porque não funcione, mas porque foi construído para entregar exatamente aquilo que se quer esconder.

Alternativas mais modernas

Se o e-mail é ruim de origem, o que usar? Depende do que você precisa. Se **precisa mesmo** de um endereço de e-mail (para cadastros, para falar com o mundo civil), há provedores que operam **serviços .onion** — a caixa de e-mail acessível por dentro da rede Tor, sem nó de saída, o que remove o vazamento de IP na ponta do acesso (ProtonMail e Mailbox.org, por exemplo, publicam endereços `.onion` para o webmail). Isso combate o **problema do IP**, mas não muda a natureza do protocolo: os metadados entre provedores continuam expostos, e a retenção continua existindo.

A distinção mais importante, porém, é conceitual: **e-mail não é mensageria**. Para conversa sensível, o instrumento certo não é um e-mail "mais seguro", e sim uma tecnologia de **mensageria** projetada desde o início para privacidade, como o **XMPP** com a extensão **OMEMO** (criptografia ponta a ponta com forward secrecy) ou o **Signal** (protocolo com sigilo futuro por padrão, metadados minimizados e servidores que guardam propositalmente quase nada). Essas ferramentas nascem com o **forward secrecy** que falta ao PGP e com um modelo de retenção mínima que o e-mail nunca terá. A regra prática que eu deixo é esta: **use o e-mail como uma identidade civil descartável e um veículo de operação; use mensageria moderna quando o assunto de fato importa; e nunca peça a uma ferramenta que ela faça o que não foi construída para fazer.**

E-mails descartáveis

Como há sempre a necessidade de um e-mail para algum processo de cadastro, entra em cena o **e-mail avulso** (ou descartável): um serviço de e-mail que não requer nenhum cadastro — simplesmente você entra num site e obtém um e-mail qualquer. Hoje temos dois serviços profissionais que valem citar:

- **YopMail** — o serviço gratuito, rápido e rico em recursos do YOPmail protege você contra spam. Em vez de expor seu e-mail real, use o descartável do YOPmail para se inscrever onde quiser.
- **EmailOnDeck** — é o principal site para tudo relacionado a conta de e-mail temporário e descartável; procura ajudar a proteger a privacidade de e-mails não avulsos, evitando SPAM.

O **YopMail** é o serviço mais conhecido e tem uma interface muito simples. O problema dele é que o **domínio é sempre o mesmo (yopmail.com)**, e muitos serviços que exigem e-mail como chave de cadastro **negam** novos cadastros com esse domínio, justamente por saberem que é descartável.

Já o **EmailOnDeck** sempre **cria novos domínios**, então por ele é mais fácil localizar um e-mail que **não** esteja na lista negra de domínios de e-mail (na verdade, nunca tive esse

problema). Na interface principal, você passa pela validação de anti-bot clássica e em seguida basta clicar em **Receber seu email**. Um e-mail é gerado randomicamente com um domínio genérico, e um **cookie** é gerado para manter a sua associação a esse e-mail. Atenção ao detalhe: caso você apague os cookies ou esteja em um browser em modo privado, esse e-mail **não poderá ser recuperado** no futuro. Existe a possibilidade de exportar uma chave de acesso, mas o serviço é pago deste ponto em diante.

Ferramentas citadas

- **Postfix** — servidor de e-mail (MTA/SMTP) para GNU/Linux, usado nos exemplos de instalação de servidor próprio; **código aberto** (licença IBM Public License / EPL).
- **ProtonMail (Proton Mail)** — provedor de e-mail com criptografia de conteúdo, sediado na Suíça e acessível também por serviço `.onion`; serviço **comercial** (freemium), com componentes de cliente em código aberto (GPLv3).
- **YopMail** — serviço web de e-mail descartável de domínio fixo (`yopmail.com`); serviço **comercial/gratuito** (proprietário).
- **EmailOnDeck** — serviço web de e-mail temporário e descartável com geração de domínios variados; serviço **comercial** (freemium, proprietário).
- **GnuPG (GNU Privacy Guard)** — implementação livre do padrão OpenPGP para cifrar e assinar mensagens e arquivos; **código aberto** (GPLv3).

Referências

- TANENBAUM, Andrew S.; WETHERALL, David J. **Redes de Computadores**. 5. ed. São Paulo: Pearson, 2011.
- KLENSIN, John. **RFC 5321 — Simple Mail Transfer Protocol (SMTP)**. IETF, 2008.
- CALLAS, Jon et al. **RFC 4880 — OpenPGP Message Format**. IETF, 2007.
- PROTON AG. **Statement about a case involving Proton Mail and a French climate activist**. Andy Yen, 2021. Disponível em: https://proton.me/blog/
- TECHNADU. **ProtonMail Logged IP Address of French Activist After Order From Swiss Authorities**. 2021.
- THE VERGE. **ProtonMail logged IP address of French activist after order from Swiss authorities**. 2021.

Download Anônimo de Torrents

Muitos arquivos no mundo hacker são expostos por uma rede descentralizada **P2P** (*peer-to-peer*), sem centralização: um arquivo rapidamente se multiplica nesta rede, levando ao cenário em que é quase que impossível parar sua divulgação. Assim foi para vários vazamentos, desde vazamento de CPF até dados do Twitch. Os links geralmente rolam em chats públicos como o 4channel.org ou privados, e para os grupos que vazaram a informação é importante o maior número de **peers** (pares) na rede com seus arquivos vazados. Este capítulo trata de como participar dessa rede sem entregar quem você é — e, principalmente, de um caminho que muita gente escolhe por engano e que precisa ser dito com todas as letras: **torrent não se baixa pela rede Tor.**

Por que este ponto é perigoso

Para um hacker, este ponto é perigoso. Embora ter dados obtidos de forma pública não seja um crime, um artifício vem sendo usado pelos juristas para facilitar a punição de quem possui tais dados: a alegação de **direitos autorais**. A partir daí, com a apreensão da máquina para busca de artefatos com direitos autorais, comprova-se por outros arquivos outras ilegalidades, segundo as leis impostas por uma ou outra máfia local. O detalhe técnico que fecha essa armadilha está na própria natureza do protocolo, e vale entendê-lo antes de tocar em qualquer cliente.

Sempre que é realizado o download de um trecho de um arquivo por P2P, é criada uma conexão entre quem está cedendo aquele trecho de arquivo e quem está baixando. Neste cenário, a conexão direta expõe as pessoas, e desta forma as máfias locais conseguem registrar possíveis ilegalidades cometidas por hackers e entusiastas. O ponto central é este: no BitTorrent **o seu IP é o dado de troca**. Para receber pedaços, você precisa dizer aos outros pares onde encontrá-lo, e isso significa anunciar um endereço para a rede inteira. Toda a estratégia de anonimato num download de torrent se resume a garantir que o endereço anunciado **não seja o seu IP real** — e é justamente aqui que quase todo mundo erra.

O erro fatal: torrent pela rede Tor

Já foi dito no capítulo do Tor, e repito aqui porque é a única falha deste capítulo capaz de queimar toda a sua proteção de uma vez: **não baixe torrents pela rede Tor.** A intuição de que "se o Tor anonimiza tudo, então também anonimiza o torrent" é falsa, e a razão é técnica, não uma questão de opinião.

O Tor transporta apenas tráfego **TCP** encaminhado por um proxy **SOCKS5**. Um cliente BitTorrent faz muito mais do que isso. Ele usa **UDP** para o **DHT** (*Distributed Hash Table*, a tabela distribuída que localiza pares sem servidor central), para o **PEX** (*Peer Exchange*, a troca de pares entre quem já está no enxame) e para tentativas de conexão direta; e o UDP simplesmente **não passa pelo proxy SOCKS do Tor** — ele sai por fora, pela sua conexão normal. Pior: mesmo o pouco que passa pelo proxy carrega, dentro da própria mensagem do protocolo, o endereço que o cliente acredita ser o seu, anunciado no campo de handshake e nas mensagens do tracker. O resultado é que o cliente **anuncia o seu IP real por fora do túnel**, e todo o circuito Onion vira teatro.

Isso não é teoria: é um ataque documentado. Em 2011, Le Blond e colegas, no estudo "*One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users*",

mostraram exatamente isso. Eles exploraram clientes BitTorrent rodando sobre Tor para descobrir o IP real do usuário — a "maçã podre" do título — e, uma vez de posse desse IP, conseguiram **correlacionar** e desanonimizar também o outro tráfego Tor daquele mesmo usuário, inclusive navegação web que nada tinha a ver com o torrent. Ou seja, um único cliente P2P mal configurado não só entrega o download: contamina toda a sessão Onion.

E há um segundo motivo, de ordem coletiva. A rede Tor é um **bem comum e escasso** — poucos milhares de relays voluntários carregam o tráfego de milhões de pessoas que dependem dele para escapar de censura e vigilância. Torrent é o tipo de tráfego mais pesado e mais contínuo que existe. Jogar downloads de arquivos grandes em cima do Tor esmaga a banda de quem precisa da rede para coisas que não podem esperar, e ainda por cima entrega o próprio usuário. É a pior combinação possível: ineficaz para você e prejudicial para todo mundo. Para torrent, portanto, existem exatamente dois caminhos corretos, e é sobre eles que trata o resto do capítulo: **VPN com bind na interface do túnel** (a rota principal, que veremos com o qBittorrent) e **I2PSnark dentro da I2P** (a alternativa quando não há VPN).

Amarrar à interface não é o mesmo que usar proxy

Antes dos comandos, é preciso fixar uma distinção que separa o download seguro do download furado, porque ela também explica por que o Tor falha e por que a VPN, feita direito, funciona.

Existem duas formas de forçar um cliente a sair por um caminho protegido. A primeira é apontá-lo para um **proxy** (SOCKS/HTTP) e torcer para que ele mande **tudo** por ali. O problema é o "tudo": como vimos, o BitTorrent gera UDP, DHT, PEX e tentativas de conexão que muitos clientes não sabem — ou não se dão ao trabalho de — empurrar pelo proxy. O que escapa vaza o IP real. É exatamente o mecanismo do ataque *bad apple*.

A segunda forma, muito mais robusta, é **amarrar o cliente a uma interface de rede** (*network interface binding*). Em vez de dizer "use este proxy", você diz "só existe uma placa de rede para você, e é a `tun0`" — a interface virtual criada pela VPN. Se a VPN cair, a interface `tun0` desaparece, e o cliente amarrado a ela fica **sem nenhuma rota**: não há vazamento porque não há por onde vazar. Isso funciona como um **kill switch** de fato (assunto do capítulo de VPN): a queda do túnel não expõe o tráfego, apenas o interrompe. É por isso que a configuração que segue não usa proxy nenhum — ela prende o qBittorrent à `tun0` e ponto. Todo pacote, TCP ou UDP, DHT ou PEX, é obrigado a nascer com o IP da VPN, porque é o único endereço que aquela interface tem.

Vale ainda desligar o **UPnP** (*Universal Plug and Play*) e o **NAT-PMP** nas opções do cliente. Esses protocolos pedem ao seu roteador que abra portas automaticamente, e nesse diálogo podem expor a topologia e o endereço da sua rede local por fora do túnel. Num download anônimo, o mapeamento de portas se faz na mão, ou não se faz.

Instalando o qBittorrent

Para este cenário será utilizado o **qBittorrent**. Caso não o tenha instalado, faça a instalação conforme a listagem abaixo:

```
  1 sudo apt-get install software-properties-common
263 sudo add-apt-repository ppa:qbittorrent-team/qbittorrent-stable
264 sudo apt-get update
265 sudo apt-get install qbittorrent
```

O primeiro pacote, `software-properties-common`, fornece o comando `add-apt-repository`; o segundo passo adiciona o **PPA** (*Personal Package Archive*) oficial da equipe do qBittorrent, para termos a versão estável mais atual em vez da que vem congelada nos repositórios da distribuição; e os dois últimos atualizam o índice e instalam o cliente.

Amarrando o qBittorrent à interface do túnel VPN

Para fazer a configuração correta, utilizando uma VPN privada, acesse **Settings** (Ferramentas → Opções) do qBittorrent. Procure, nas opções avançadas (**Advanced**), a opção de escolha da interface na qual está sendo utilizado o tunelamento VPN. Na figura abaixo temos a `tun0` como a interface virtual que está respondendo pelo tunelamento — é ela que devemos selecionar no campo *Network Interface* (Interface de rede).

Simples, mas antes de usar, deve-se confirmar que a configuração está operacional. Para isso, clique em **View** (Visualizar) e escolha a opção **Log** (Registro de execução), conforme a figura abaixo.

Quando exibir o Log (Show), veja na figura abaixo que ele é categórico: **"Detected external IP: 154.13.1.155"**. Ou seja, é um IP em outro país, o que corrobora o uso da VPN. Esse é o teste que fecha a configuração: se o IP externo detectado fosse o da sua conexão doméstica, seria sinal de que o cliente não está saindo pela `tun0` e você **não deve** iniciar nenhum download.

Sempre esteja atento ao Log. Veja abaixo que é possível consultar na Internet a **geolocalização** de um IP: basta pegar o endereço que o próprio qBittorrent detectou e conferir, num serviço de consulta de IP, que o país corresponde ao servidor da sua VPN — e não ao seu. Faça essa conferência a cada sessão, porque servidores de VPN mudam de endereço e um túnel que caiu e voltou pode ter reconectado por outra rota.

Alternativa sem VPN: TOR ou I2P

Caso não tenha um serviço privado de VPN, o manuscrito aponta o TOR ou a **I2P** como alternativas. Aqui é preciso corrigir e detalhar, para não cair na armadilha que abriu este capítulo: para **navegação** anônima, o Tor é excelente; para **torrent**, ele é o caminho errado, pelos motivos já expostos (o cliente anuncia o IP real por fora do proxy, e o estudo *bad apple* mostra que isso desanonimiza até o resto da sua sessão). Portanto, quando não há VPN, a alternativa correta para P2P **não** é rodar o qBittorrent sobre Tor — é usar a **I2P**.

A **I2P** (*Invisible Internet Project*) é uma rede anônima que, ao contrário do Tor, foi desenhada de dentro para fora pensando em tráfego **par a par** dentro dela mesma. Em vez de sair para a internet comum (como o Tor faz por seus nós de saída), a I2P mantém o tráfego **dentro** da rede, roteado por túneis unidirecionais separados para entrada e saída. Ela traz embutido um cliente de torrent próprio: o **I2PSnark**. Como o I2PSnark só fala com pares que também estão dentro da I2P, e como o transporte inteiro do arquivo acontece por túneis I2P, não existe o vazamento de UDP/DHT "por fora" que condena o BitTorrent sobre Tor — o DHT e a descoberta de pares são a própria rede anônima, não um canal paralelo que escapa dela. O preço é que você fica restrito aos torrents que circulam dentro da I2P, tipicamente mais lentos e menos populosos que o enxame da internet aberta. É uma troca consciente: menos velocidade e menos catálogo em nome de um anonimato que não depende de você configurar bind e kill switch corretamente.

Resumindo a decisão em uma frase: com VPN, use o **qBittorrent amarrado à `tun0`** com UPnP desligado; sem VPN, use o **I2PSnark dentro da I2P**; e **nunca** o BitTorrent sobre Tor.

Ferramentas citadas

- **qBittorrent** — cliente BitTorrent multiplataforma, leve, sem anúncios, com opção de fixar a interface de rede de saída; **código aberto** (licença GPL v2+).
- **I2PSnark** — cliente de torrent integrado ao roteador I2P, que mantém todo o tráfego P2P dentro da rede anônima I2P; **código aberto** (parte do projeto I2P, licenças livres — majoritariamente domínio público / BSD / GPL conforme o componente).

Referências

- LE BLOND, Stevens; MANILS, Pere; CHAABANE, Abdelberi; KAAFAR, Mohamed Ali; CASTELLUCCIA, Claude; LEGOUT, Arnaud; DABBOUS, Walid. **One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users**. In: 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), Boston, 2011.
- COHEN, Bram. **The BitTorrent Protocol Specification (BEP 3)**. BitTorrent.org, 2008.
- LOEWENSTERN, Andrew; NORBERG, Arvid. **DHT Protocol (BEP 5)**. BitTorrent.org, 2008.
- I2P PROJECT. **I2PSnark and the I2P Network — Technical Documentation**. Disponível em: https://geti2p.net/
- QBITTORRENT PROJECT. **qBittorrent official documentation — Advanced settings: Network Interface binding**. Disponível em: https://www.qbittorrent.org/

As Recomendações de Edward Snowden

Poucos nomes carregam tanto peso na história recente da privacidade quanto o de Edward Snowden. Em junho de 2013, ainda contratado da NSA através da Booz Allen Hamilton, ele entregou a jornalistas do **The Guardian** e do **The Washington Post** um arquivo de documentos que provou, com papel timbrado, algo que até então era tratado como paranoia de gente de chapéu de alumínio: os Estados Unidos operavam um sistema de **vigilância em massa** sobre as próprias comunicações do planeta. Programas como o **PRISM** coletavam dados diretamente dos servidores das grandes empresas de tecnologia; a coleta de metadados telefônicos varria milhões de pessoas sem suspeita individual nenhuma. Snowden virou foragido, refugiou-se em Moscou, e anos depois contou tudo em suas próprias palavras no livro **"Permanent Record"** (2019, publicado no Brasil como "Eterna Vigilância"). Quando um homem que trabalhou por dentro da maior máquina de espionagem já construída se dá ao trabalho de recomendar ferramentas, vale a pena ouvir — e é isso que este capítulo faz, com as minhas ressalvas de sempre.

Comunicar-se cifrado

Primeiro, precisa se comunicar com outros e naturalmente em um ambiente criptografar toda comunicação. O leitor pode usar para isso um aplicativo chamado **Signal** da Open Whisper Systems, trata-se de um aplicativo gratuito. Qualquer pessoa pode usar, mas caso não queira ter um celular, o que acho até certo, pode-se usar **XMPP** (*Extensible Messaging and Presence Protocol*) com softwares de comunicação que criptografam. Acredito que o texto seja a melhor forma de se comunicar, ou seja, evita-se áudio. O disco deve ser criptografado, afinal o computador pode ser roubado pelo estado para saber o que anda fazendo. Então como está criptografado nada pode ser usado contra você. Você deve acreditar que apenas fotos e áudio não são importantes para o estado, mas garanto, é sim.

Aqui cabe uma atualização honesta que o tempo impôs ao texto original. A **Open Whisper Systems**, a organização de Moxie Marlinspike que criou o Signal, foi dissolvida em 2018; hoje o aplicativo é mantido pela **Signal Foundation**, uma fundação sem fins lucrativos, com o mesmo protocolo aberto de sempre — o *Signal Protocol*, cuja criptografia de ponta a ponta é tão bem-feita que foi licenciada por WhatsApp, Google e outros. Ou seja, a recomendação continua de pé; só mudou o nome de quem cuida da casa. E a preferência pelo texto no lugar do áudio, que defendo acima, tem fundamento técnico: áudio e vídeo carregam metadados, biometria de voz e um volume de dados que amplia a superfície de análise. Escrito, cifrado e efêmero é sempre menos rastro.

Disco cifrado e a lógica do confisco

O ponto do disco criptografado merece ser martelado, porque muita gente ainda o subestima. Quando escrevo que "nada pode ser usado contra você", não é força de expressão: um disco cifrado com uma senha forte, desligado no momento da apreensão, é uma parede. O erro fatal — e voltaremos a ele quando falarmos de opsec — é ser pego com a máquina **ligada e destravada**, porque aí a chave está na memória e toda a criptografia vira decoração. Presuma que o computador pode ser levado. Presuma que fotos e áudio "sem importância" são, sim, importantes para quem monta um perfil sobre você.

A ressalva sobre gerenciadores de senha

Edward Snowden recomenda uso de **gerenciador de senhas** (*password manager*). Uma das principais coisas que expõem as informações privadas das pessoas, não necessariamente aos adversários mais poderosos. Suas credenciais podem ser reveladas porque algum serviço que você parou de usar em 2007 foi hackeado, e a senha que você usava para aquele site também funciona para sua conta do Gmail. Um gerenciador de senhas permite que você crie senhas exclusivas para cada site que sejam inquebráveis, mas você não tem o fardo de memorizá-las. Há controvérsias de minha parte, gerenciadores de senha local e remotos são invadidos todos os anos, o autor deste livro neste ponto prefere confiar na cabeça.

Mantenho a minha desconfiança, mas seria desonesto não apresentar o meio-termo que amadureceu nos últimos anos. Um bom cofre **de código aberto e local** — que nunca sincroniza na nuvem de terceiros — resolve boa parte da minha objeção, porque tira o alvo do servidor remoto de uma empresa. O caso mais citado é o **KeePassXC**: um banco de senhas cifrado que fica num arquivo `.kdbx` **na sua própria máquina**, sem conta, sem nuvem, sem servidor de ninguém para invadir. A diferença em relação aos gerenciadores "todos os anos invadidos" é justamente essa: o vazamento em massa acontece quando há um servidor central guardando milhões de cofres. Sem esse servidor, o adversário teria que atacar você, individualmente, e ainda quebrar a sua senha-mestra. Quem, como eu, prefere confiar na cabeça pode adotar um híbrido: memorizar a senha-mestra e as poucas contas que realmente importam, e delegar ao KeePassXC apenas o resto — as centenas de logins descartáveis que ninguém consegue guardar sem repetir. Repetir senha é o pecado que Snowden descreve; um cofre local é a penitência menos arriscada.

Dois fatores, e por que prefiro chave física a celular

A outra coisa é a **autenticação de dois fatores** (*two-factor authentication*, 2FA). O valor disso é se alguém roubar sua senha, ou ela for deixada ou exposta em algum lugar, a autenticação de dois fatores permite que o provedor envie a você um meio secundário de autenticação. Se você habilitar a autenticação de dois fatores, um invasor precisará de sua senha como primeiro fator e de um dispositivo físico, como seu telefone, como segundo fator, para fazer login em sua conta. Gmail, Facebook, Twitter, Dropbox, GitHub, Battle.net e muitos outros serviços suportam autenticação de dois fatores. O autor desta obra discorda apenas no uso de celular, se puder optar por uso de chaves físicas como **Yubico**, faça, mas tenha muitas pois evite usar as mesmas chaves entre contas do seu eu real e do seu eu hacker (papo de doido).

Vale explicar por que a chave física é melhor do que o SMS ou o app do celular, porque não é frescura. O segundo fator por telefone tem duas fraquezas conhecidas: o **SIM swap**, em que o atacante convence a operadora a portar seu número para um chip dele e passa a receber seus códigos, e o **phishing**, em que um site clonado captura o código de seis dígitos que você digita e o repassa em tempo real. A chave física da linha **YubiKey** (fabricada pela Yubico) fecha essas duas portas: ela implementa o padrão **FIDO2/WebAuthn**, no qual a autenticação é **amarrada criptograficamente ao domínio real** do site — se o endereço não for exatamente o verdadeiro, a chave simplesmente não responde, e nenhum código trafega para ser roubado. Some a isso o meu conselho de compartimentar: chaves separadas para o "eu real" e para o "eu hacker", nunca as mesmas entre os dois mundos, para que um vazamento jamais costure as duas identidades. Não é papo de doido; é higiene de identidade.

Adotar tecnologia confiável aos poucos

Uma observação importante feita por ele é que ao longo de nosso dia a dia temos que nos cercar de tecnologias que podemos confiar e que pode-se fazer aos poucos, pois muda o estilo de vida. No caso do autor deste livro, estou passando por este quanto ao pacote Google Docs, estou migrando para **Disroot.org** — uma plataforma sem fins lucrativos, hospedada na Holanda, que oferece nuvem, e-mail e editor colaborativo sobre software livre, sem o modelo de negócio da vigilância. É por isso que Edward gosta de aplicativos como o Signal, porque possuem boa curva de aprendizado. Não exige que você reordene sua vida. Não exige que você altere seu método de comunicação. Essa gradualidade é a chave: quem tenta trocar tudo num fim de semana desiste na segunda-feira. Troca-se uma peça, ela vira hábito, troca-se a próxima.

O sistema operacional de Snowden: compartimentalização

Sobre sistemas operacionais, Edward sempre utiliza sandbox virtualizadas com **Qubes OS**. Aqui o texto original merece um aprofundamento, porque essa escolha não é aleatória e liga direto ao capítulo do Whonix. O Qubes OS parte de um princípio chamado **compartimentalização** (ou *security by compartmentalization*): em vez de um único sistema onde um aplicativo comprometido contamina tudo, o Qubes divide o computador em várias máquinas virtuais isoladas — os *qubes* —, cada uma com um propósito e um nível de confiança. Um qube para trabalho, outro para banco, outro descartável para abrir um anexo suspeito. Se um cair, os outros continuam de pé. E, crucialmente, Snowden combina o Qubes com o **Whonix**: toda a navegação passa por um qube que só fala pela rede Tor, enquanto o navegador roda em **outro** qube que não conhece o seu IP real. É a mesma separação que este livro defende no capítulo do Whonix — separar quem navega de quem tem endereço na rede — só que elevada a filosofia de sistema operacional inteiro. A lição do caso Freedom Hosting, que vimos antes, é exatamente essa: uma brecha no navegador não pode, sozinha, anular toda a proteção de rede.

As ferramentas que Snowden cita

Quanto aos *browsers*, Edward afirma que **TOR** é fundamental e que também um bom browser ajuda. Recomenda-se o uso de ferramentas como **HTTPS Everywhere** e *ad-blockers*. Em seu livro ele apresenta a lista abaixo, que reproduzo com as descrições dele:

Signal Messenger

Signal é um aplicativo de smartphone de mensagens instantâneas criptografadas. Eu sei o que você está pensando: se mantiver atualizado com os patches de segurança do seu sistema operacional, o Signal ainda permanece útil. Os dados do WikiLeaks não mostram que aplicativos de bate-papo criptografados foram quebrados. E se o seu sistema operacional é *buggy*, nenhum nível de criptografia pode ajudar de qualquer maneira.

TOR

TOR significa **The Onion Router**, que basicamente adiciona camadas às solicitações que você está enviando através do seu navegador. Então, quando você pede ao seu navegador para abrir um site, a solicitação não vai direto para o site, ela é encaminhada por vários países diferentes para que sua localização permaneça privada. Jornalistas, ativistas e aqueles que usam a internet de países onde a liberdade de expressão é restrita usam o TOR regularmente para segurança, não apenas para a privacidade. Mas, sim, o TOR também se

presta a acessar a darknet e a deep web, que é uma outra história — e é a história do capítulo dedicado ao Onion Router neste livro.

Tails

Falando em TOR, o sistema operacional que Snowden recomenda é o código aberto **Tails OS** ou *The Amnesic Incognito Live System*. O nome diz tudo. Literalmente. A ideia por trás do Tails é que ele funciona a partir de uma unidade USB ao vivo e não deixa nenhuma pegada digital na máquina em que você a conecta. O sistema operacional permite que você use a internet de forma anônima e usa ferramentas de criptografia para seus arquivos, e-mails e outros usos. Instalá-lo é complicado, porém, tornando isso estritamente uma ferramenta para o usuário mais paranoico ou para aqueles que mais precisam de segurança e privacidade.

HTTPS Everywhere

De todas as ferramentas desta lista, **HTTPS Everywhere** é provavelmente a mais fácil de usar. É uma extensão de navegador que você pode facilmente adicionar ao Chrome, Firefox ou Opera. Em termos simples, o HTTPS permite sua comunicação através de uma rede através de um canal seguro. É apenas uma camada adicional de segurança sobre o **Hypertext Transfer Protocol** ou HTTP. Então, se um site o suporta, essa extensão fará com que o site use automaticamente o HTTPS, tornando sua navegação mais segura.

Aqui entra a atualização mais importante deste capítulo, e faço questão de registrá-la para não deixar o leitor com informação vencida. O **HTTPS Everywhere foi oficialmente descontinuado** — a EFF e o Tor Project encerraram a extensão em 2022 e ela chegou ao fim de vida em 2023. O motivo é a melhor das razões: **a extensão venceu a sua própria batalha**. Quando ela nasceu, em 2010, boa parte da web ainda trafegava em HTTP puro e forçar o HTTPS era exceção; hoje o HTTPS virou o padrão da web, e os próprios navegadores incorporaram nativamente o comportamento que a extensão oferecia. Firefox, Chrome, Edge e Brave já trazem um modo **"HTTPS-Only" / "Always Use Secure Connections"** embutido nas configurações. Portanto, não instale mais o HTTPS Everywhere: em vez disso, ligue o modo somente-HTTPS do seu navegador. A recomendação de Snowden continua correta no espírito; só migrou da extensão para dentro do próprio browser.

SecureDrop

E se você é um jornalista ou um tipo de denunciante inspirado em Edward Snowden, então essa ferramenta é o que você está procurando. **SecureDrop** é uma plataforma que facilita a comunicação com segurança através da rede TOR. Os sites SecureDrop só são acessíveis na rede TOR, portanto. Não por acaso, o SecureDrop é hoje mantido pela **Freedom of the Press Foundation**, cujo conselho Snowden integra — é a materialização, em código, do canal seguro que teria facilitado a vida dele em 2013.

Ad-blockers (não é brincadeira)

Mas tenha coragem, porque se você não se vê usando nenhuma dessas ferramentas, Snowden tem mais uma sugestão: use um **bloqueador de anúncios** (*ad-blocker*). E sabemos que a maioria de vocês usa um desses. Os provedores de serviços que apresentam anúncios com conteúdo ativo exigem JavaScript para exibir e precisam de algo como o Flash incorporado neles, e todos eles podem ser usados para atacar seu navegador da web. Um bloqueador de anúncios tenta ativamente detê-los. Não é sobre estética nem sobre poupar dados: é redução de superfície de ataque. Cada script de terceiro que não carrega é um vetor a menos.

Um mapa das recomendações

Reunindo tudo, o conselho de Snowden não é uma lista solta de aplicativos, e sim camadas que se reforçam: comunicação cifrada por baixo, um sistema operacional compartimentado por cima, e a navegação escondida na rede Tor no meio. A figura a seguir organiza essas camadas — da mensagem que você digita até o hardware que a executa —, com as minhas ressalvas anotadas onde discordo do original.

Ferramentas citadas

- **Signal** — mensageiro com criptografia de ponta a ponta (*Signal Protocol*), mantido pela Signal Foundation; **código aberto** (GPLv3 / AGPLv3).
- **Tor** — rede de roteamento anônimo (roteamento Onion); **código aberto** (licença BSD de 3 cláusulas).
- **Tails** — sistema operacional *live* e amnésico que roteia tudo pelo Tor; **código aberto** (GPLv3).
- **SecureDrop** — plataforma de submissão anônima para jornalistas via Tor, mantida pela Freedom of the Press Foundation; **código aberto** (AGPLv3).
- **Qubes OS** — sistema operacional baseado em compartimentalização por máquinas virtuais (Xen); **código aberto** (GPLv2), integra o Whonix para roteamento Tor.
- **KeePassXC** — gerenciador de senhas local, sem nuvem, com cofre cifrado `.kdbx`; **código aberto** (GPLv3).
- **YubiKey** — chave física de segundo fator (FIDO2/WebAuthn, U2F, OTP), fabricada pela Yubico; **hardware comercial** (com bibliotecas de suporte de código aberto).

Referências

- SNOWDEN, Edward. **Permanent Record**. New York: Metropolitan Books, 2019. (Ed. brasileira: **Eterna Vigilância**. São Paulo: Planeta, 2019.)
- GREENWALD, Glenn; MACASKILL, Ewen. **NSA Prism program taps in to user data of Apple, Google and others**. The Guardian, 7 jun. 2013.
- GREENWALD, Glenn. **NSA collecting phone records of millions of Verizon customers daily**. The Guardian, 6 jun. 2013.
- ELECTRONIC FRONTIER FOUNDATION. **HTTPS Everywhere Now Deprecated as HTTPS Adoption Reaches Majority**. 2021–2022.
- FREEDOM OF THE PRESS FOUNDATION. **SecureDrop**. Disponível em: https://securedrop.org/
- QUBES OS PROJECT. **Qubes OS: A reasonably secure operating system**. Disponível em: https://www.qubes-os.org/

Password e Username

Toda autenticação pede duas coisas: quem você é e o segredo que só você deveria saber. O **username** e o **password**. Parecem detalhes de cadastro, coisa de tela de login, mas são o ponto exato onde a maioria das invasões acontece — não por gênio criptográfico, e sim porque as pessoas escolhem senhas ruins, repetem senhas entre serviços e usam nomes de usuário previsíveis. Neste capítulo trato dos dois lados da moeda: como um atacante descobre usuário e senha, e como eu me defendo — inclusive com uma chave física que digita por mim caracteres que eu nem sei quais são. É um capítulo tanto de ataque quanto de higiene pessoal, porque nesse assunto quem não se defende com método vira estatística.

A senha que você nunca deveria reutilizar

Os passwords devem ser fortes, ou seja, números, letras minúsculas e letras maiúsculas, não esqueça dos caracteres especiais. Nunca use a mesma senha em tudo, use variações, infelizmente é complicado. Eu utilizo um hardware próprio associado a alguns caracteres extras, então fica complexo.

Para alguns sites idiotas uso algumas senhas, já para sites importantes e acesso a sistemas operacionais, senhas tão fortes que é comum eu errar 5 a 9 vezes ao dia. Caso seja curioso e queira ver o que não se deve fazer, procure ver listas de passwords. Passwords podem ser obtidos de várias formas, a principal é obter listas de prováveis senhas em:

- Sites que possuem **Data Breach** de serviços web com senhas vazadas;
- Sites onde as pessoas testam suas senhas (sim, existe gente que fica testando — informando — a própria senha em um site qualquer da internet, e aquilo vira lista).

Também é possível obter senhas em sites hacker mais privados ou em grupos, principalmente no Telegram. Um bom arquivo de senhas tem como primeiro elemento o valor `123456`, e entre os dez primeiros estão sempre os mesmos suspeitos:

```
  1 123456
266 password
267 qwerty
```

A lista completa das piores muda pouco de ano para ano — depois de `123456` e `password` vêm `123456789`, `12345678`, `123123`, `1234567890`, `qwerty`, `abc123`, `1q2w3e` e assim por diante. Elas aparecem no topo de todo relatório de senhas vazadas justamente porque milhões de pessoas as escolhem, e é isso que as torna as primeiras a serem testadas por qualquer ataque.

Por que reutilizar senha é o pecado capital

Antes de falar de ataque, preciso reforçar uma ideia que já apareceu no capítulo sobre o Snowden e que aqui fica ainda mais concreta: a senha reutilizada é o que transforma um vazamento pequeno em um desastre pessoal. Quando um site qualquer é invadido e o banco de senhas vaza, o atacante não fica preso àquele site. Ele pega o par `e-mail + senha` e testa **automaticamente** em dezenas de outros serviços — e-mail, banco, rede social, painel de nuvem. Isso se chama **credential stuffing** (recheio de credenciais): não se quebra nada, apenas se reaproveita o que a vítima entregou de graça em outro lugar. Se você usa a mesma senha no fórum idiota e no seu e-mail principal, o dia em que o fórum vazar é o dia em que você perde o e-mail.

É por essa razão que existe o **Have I Been Pwned** (HIBP), o serviço mantido por Troy Hunt que agrega bilhões de credenciais de vazamentos conhecidos. Você digita seu e-mail e ele diz em quais vazamentos aquele endereço apareceu. O mesmo projeto expõe o **Pwned Passwords**, uma base de centenas de milhões de senhas já vazadas que pode ser consultada de forma segura — o serviço usa *k-anonymity*: você envia só os cinco primeiros caracteres do hash SHA-1 da sua senha e recebe de volta a lista de sufixos correspondentes, comparando localmente, de modo que a senha inteira nunca sai da sua máquina. Do lado defensivo, serve para saber se uma senha sua já é conhecida. Do lado ofensivo, é uma das fontes de lista mais ricas que existem — e por isso aparece adiante entre as fontes de senhas prováveis.

A estratégia de ataque em três níveis

Infelizmente muitas listas de senhas são geradas sem nenhuma inteligência e, por isso, o processo de ataque por listas é demorado e pode ser frustrado por um administrador atento à sua infraestrutura. O recomendado é que o ataque siga três passos, do mais cirúrgico ao mais bruto:

1. Gerar uma lista de cerca de 100 senhas focadas no alvo/pessoa;
2. Obter uma lista de senhas prováveis de um site;
3. Usar uma lista gerada sem nenhuma inteligência (força bruta).

Nível 1 — as 100 senhas do alvo. Para gerar uma lista de 100 senhas focadas na pessoa, deve ser feita uma engenharia social nas redes sociais, um estudo profundo: nome do cachorro, time, data de nascimento, nome do filho, banda favorita, placa do carro, tudo aquilo que a pessoa transforma em senha achando que ninguém liga os pontos. O ataque com esta lista deve ser feito 1 por 1, com intervalos longos, de 5 a 10 minutos entre cada tentativa, para não despertar suspeita. E sim: ataques de senha devem ser **furtivos**. Cem tentativas com dez minutos de intervalo é uma tentativa a cada dez minutos — invisível no meio do tráfego legítimo, ao contrário de cem tentativas por segundo, que qualquer sistema de detecção acende como árvore de Natal.

Nível 2 — as senhas prováveis do serviço. Se as 100 focadas falharem, parte-se para um arquivo de senhas prováveis já ordenadas por frequência. Recomendo consultar fontes como:

- **Pwned Passwords** (Have I Been Pwned);
- **Top 200 Most Common Passwords** da NordPass, atualizada todo ano;
- Coletâneas públicas de senhas vazadas, do tipo `password_2021-06-16` disponíveis para download em arquivos abertos.

Como aqui existem muitas senhas, não dá para atacar de forma tão pausada quanto no nível 1; o ataque passa a ser massivo e, portanto, de fácil detecção. É a troca clássica: quanto mais tentativas, mais chance de acerto e mais barulho.

Nível 3 — a força bruta. Se mesmo assim não deu, é a hora da força bruta: gerar arquivos com senhas de 8 a 16 caracteres em sequência e executar tudo contra o alvo. É o método mais caro e mais ruidoso, e só se justifica quando o alvo é valioso e os níveis anteriores fracassaram.

Entropia: por que comprimento vence complexidade

Vale abrir um parêntese técnico que muda a forma de pensar em senha, tanto para atacar quanto para defender. A força real de uma senha não se mede por "ter símbolo e número", e sim por **entropia** — a quantidade de incerteza, medida em bits, que um atacante precisa

vencer. Cada bit de entropia dobra o espaço de busca. Uma senha de 8 caracteres "complexa" como `P@ss1!23` tem menos entropia do que parece, porque humanos seguem padrões previsíveis (maiúscula no começo, número e símbolo no fim) que as ferramentas de ataque já conhecem e testam primeiro.

O caminho que rende mais entropia por esforço de memória é o **comprimento**, não a complexidade decorada. Daí a ideia da **passphrase** gerada por **diceware**: sorteiam-se, com dados de verdade ou com um gerador confiável, cinco ou seis palavras aleatórias de uma lista — algo como `correto-cavalo-bateria-grampo-vulcão`. É trivial de lembrar e, ainda assim, tem entropia altíssima porque o número de combinações de seis palavras entre milhares é astronômico. Aquela tirinha famosa do xkcd resumiu tudo: passamos vinte anos treinando gente para criar senhas difíceis de humanos lembrarem e fáceis de computadores adivinharem. Comprimento vence complexidade — grave isso.

Descobrindo o username

Mas em uma autenticação são precisas duas informações, o usuário e a senha. Ambas são um problema para o hacker; a diferença é que, para o usuário, existem meios de descobrir:

- **Prováveis usuários da tecnologia:** `root` no Linux, `Administrator` no Windows e `admin` em roteadores caseiros. São padrões de fábrica que muita gente nunca troca.
- **Prováveis usuários de uma empresa:** geralmente uma empresa tem uma **máscara** para criar contas, do tipo `SOBRENOME-NOME` (ou `nome.sobrenome`, `nsobrenome`, e por aí vai). Descoberto o padrão de um funcionário — coisa que um e-mail de contato no site já entrega —, você deriva o login de todos os outros.
- **Enumeração de usuários:** existem serviços vulneráveis que permitem testar se um usuário existe ou não. O próprio **SSH Server** em versões inferiores à 7.4 responde de forma mensurável — o tempo de resposta muda conforme o usuário exista ou não —, e isso permite enumerar contas válidas.

Uma forma simples de fazer essa enumeração é com o Metasploit (haverá um capítulo à parte com explicação mais densa), passando uma lista de possíveis usuários para o módulo de enumeração de SSH e deixando ele confirmar quais existem. Para um hacker com conhecimento, os caminhos podem ser traçados rapidamente — se o oponente não pensa em segurança — e o fardo árduo de uma invasão passa a ser um processo trivial. Existem inúmeras ferramentas: das mais triviais, que geram a partir do alfabeto milhões de combinações, às que geram a partir de dicionários de palavras da língua/cultura, ou até dicionários específicos para um alvo. E há ainda as que procuram a senha a partir de **hashes** localizados — afinal, um sistema não guarda a senha em texto puro, e sim criptografada por uma **One-Way-Function** (função de mão única; ver o capítulo de criptografia).

Hydra: quebrando a autenticidade, não a senha

Hydra é uma ferramenta usada para avaliar se um password é válido para usuários de um sistema. Veja que, quando ela é definida na literatura como "cracker de password", ela não quebra o password: ela quebra a **autenticidade** de um indivíduo. É uma ferramenta muito poderosa que, com ataques paralelos por *threads*, consegue realizar uma grande quantidade de testes em poucos minutos — ou seja, é um ataque **online**, contra o serviço vivo, tentando login após login.

Atualmente ela possui vários módulos, é facilmente estendida e, naturalmente, suporta uma grande quantidade de protocolos, tais como Cisco, FTP, HTTP (FORM POST e GET), IRC,

LDAP, MYSQL, MS-SQL, ORACLE, POSTGRESQL, POP/SMTP, SOCKS5, SSH, SUBVERSION, TELNET, VMWARE e XMPP.

Mas antes de atacar um alvo, deve-se mapear com **NMAP** quais serviços estão disponíveis e qual protocolo cada um fala, para a escolha correta na ferramenta. Não é preciso ter um Kali GNU/Linux para usar o Hydra: a instalação e a configuração são triviais e ele opera bem em qualquer GNU/Linux.

```
sudo apt update -y
sudo apt install hydra -y
```

John the Ripper e hashcat: quebrando o hash

O Hydra bate na porta do serviço. Já quando você tem o **hash** da senha em mãos — porque conseguiu vazar o arquivo `/etc/shadow`, o banco de um site, um handshake WPA — o jogo é outro: é um ataque **offline**, na sua própria máquina, sem precisar do alvo online e sem nenhum limite de tentativas por minuto. É aí que entram o John the Ripper e o hashcat.

John the Ripper é uma ferramenta projetada para ajudar os administradores de sistemas a encontrar senhas fracas (fáceis de adivinhar ou decifrar por força bruta) e até enviar e-mails automáticos aos usuários avisando-os sobre isso, se desejado. Ele suporta centenas de tipos de hash e cifras, incluindo: senhas de usuário de tipos Unix (Linux, *BSD, Solaris, AIX, QNX, etc.), macOS, Windows, "aplicativos web" (por exemplo, WordPress), groupware (por exemplo, Notes/Domino) e servidores de banco de dados (SQL, LDAP, etc.); capturas de tráfego de rede (autenticação de rede Windows, WiFi WPA-PSK, etc.); chaves privadas criptografadas (SSH, GnuPG, carteiras de criptomoedas, etc.); sistemas de arquivos e discos (arquivos `.dmg` do macOS e "sparse bundles", Windows BitLocker etc.); arquivos compactados (ZIP, RAR, 7z) e documentos (PDF, Microsoft Office, etc.).

```
sudo apt update -y
sudo apt install john -y
```

Hoje, porém, a ferramenta mais usada para essa tarefa não é mais o John, e sim o **hashcat**. A diferença é a **GPU**: o John, na sua forma clássica, trabalha na CPU; o hashcat foi desenhado para rodar em placas de vídeo, e uma GPU testa hashes em uma ordem de grandeza que a CPU não alcança — milhões a bilhões de tentativas por segundo, dependendo do algoritmo e do hardware. Ele suporta os principais modos de ataque (dicionário, dicionário com regras de mutação, combinação, máscara para força bruta dirigida e híbrido) e praticamente todos os formatos de hash que você vai encontrar. Na prática o fluxo é: identifica-se o tipo de hash, escolhe-se o modo (`-m` para o tipo de hash, `-a` para o modo de ataque) e aponta-se para um wordlist ou uma máscara.

```
sudo apt install hashcat -y
# Exemplo: hash NTLM (-m 1000) por dicionário (-a 0)
hashcat -m 1000 -a 0 hashes.txt rockyou.txt
```

Isso reforça algo importante sobre o nível 3 do ataque: a força bruta de 8 a 16 caracteres é inviável na mão, mas contra um hash rápido e uma boa GPU ela deixa de ser teoria. Por isso o comprimento e a entropia da senha importam tanto — cada caractere a mais multiplica o tempo de quebra.

Salt: por que as rainbow tables morreram

Falta explicar por que nem sempre o atacante ganha só por ter o hash. Se um sistema guardasse simplesmente `hash(senha)`, duas pessoas com a mesma senha teriam o mesmo hash, e um atacante poderia pré-calcular uma tabela gigante de `senha → hash` uma única vez e reusá-la contra qualquer vazamento. Essas tabelas pré-computadas se chamavam **rainbow tables** e, por um tempo, foram o terror dos hashes de senha.

O que as matou foi o **salt**: um valor aleatório, único para cada usuário, concatenado à senha antes do hash — guarda-se `hash(salt + senha)` junto com o salt. Como o salt é diferente para cada conta, a mesma senha gera hashes diferentes em usuários diferentes, e a tabela pré-computada perde o sentido: o atacante teria que refazer a tabela inteira para cada salt. Isso obriga o ataque a ser feito senha por senha, ali na hora, que é exatamente o cenário caro do hashcat. Somam-se a isso as funções de hash **lentas e propositalmente custosas** feitas para senha — bcrypt, scrypt, Argon2 —, que embutem salt e um fator de custo ajustável para deixar cada tentativa lenta de propósito. Contra um `md5(senha)` cru, a GPU voa; contra Argon2 com salt, ela engasga. É a diferença entre um banco de senhas ser um desastre em segundos ou resistir por anos.

A chave física: 32 caracteres que eu não sei

Alguns programas são cruciais para a existência do hacker e é natural que exijam algum tipo de senha ou chave de descriptografia. Os agentes maliciosos que querem capturar o hacker vão tentar obter essa senha/chave. Se você tiver que decorar 32 caracteres aleatórios, terá um problema: é uma chave grande demais para memorizar, e muita gente acaba escrevendo num arquivo TXT — o que é entregar o ouro. Mas se, para não esquecer, você usar só 12 caracteres, bom, será um alvo fácil. E um atacante virá por dois caminhos:

- **Físico:** tentará chegar até você para lhe tomar algo;
- **Virtual:** tentará acessar seus recursos e obter essa chave.

Eu observei que fisicamente é mais difícil, então bolei um esquema para uma chave física digitar por mim 32 caracteres que eu não sei, e eu digito mais 16 caracteres que eu sei. Se a chave física for roubada fisicamente, vão faltar os 16 caracteres que só eu sei. Virtualmente não tem como chegar na chave física de 32 caracteres, mas pode-se chegar até mim para descobrir os 16. Isso complica muito a vida do atacante, porque os dois caminhos, sozinhos, são incompletos: o segredo é dividido entre uma coisa que eu **tenho** (o hardware) e uma coisa que eu **sei** (os 16 caracteres).

Todo o hardware é muito simples. Comprei vários **Digispark** com Arduino e consegui um *push-button* de um computador antigo. Soldei dois pinos: se eu pressiono o botão, o aparelho digita 32 caracteres. O meu está gasto de tanto uso.

Isto é um ataque HID/BadUSB

Antes de mostrar o código, preciso nomear o que essa chave física é do ponto de vista técnico, porque é a mesma família de ataque que você usará ofensivamente. O Digispark, quando programado com a biblioteca `DigiKeyboard`, se apresenta ao computador como um **teclado USB** — um dispositivo **HID** (*Human Interface Device*). O sistema operacional não vê um pendrive nem pede permissão: vê um teclado, e teclado é confiável por definição. Então o aparelho simplesmente "digita" o que foi programado, numa velocidade sobre-humana. Essa técnica se chama **BadUSB**, e o brinquedo mais famoso dela é o **USB Rubber Ducky** da Hak5 — um pendrive que, na verdade, é um teclado que despeja comandos assim que é conectado.

No meu caso eu uso esse poder a meu favor: em vez de injetar comandos maliciosos, o dispositivo injeta os 32 caracteres da minha chave. Mas entenda que é a mesma arma. Ofensivamente, um HID desses plugado num computador destravado por dez segundos abre um terminal, baixa um payload e some — e nenhum antivírus de porta USB reclama, porque para o sistema aquilo foi só alguém "digitando muito rápido". Saber que a defesa e o ataque são o mesmo hardware é o tipo de percepção que este livro quer que você tenha.

Montando a chave no Arduino

A próxima parte recomendo que não seja feita no seu computador de uso diário. Eu tenho um notebook velho só para isso e, toda vez que faço uma chave, formato o disco com criptografia **LUKS** depois, criptografando o disco inteiro com uma chave que eu não sei — o objetivo é apagar de vez o código que digitamos no Arduino, para que os 32 caracteres não fiquem em lugar nenhum. Vamos começar instalando o Arduino no Debian:

```
sudo apt install arduino -y
```

O próximo passo é configurar o Arduino para programar essa placa com recursos Digispark. O problema é que, na época em que escrevo este livro, a biblioteca foi descontinuada; recomendo que no futuro você procure uma alternativa melhor — vou apenas mostrar como fazia. Na janela de configuração de *Boards Manager* você adiciona uma URL de índice de placas. Eu uso um repositório oficial, mas você deve procurar seus repositórios mais confiáveis; lembre-se de que há muitos ataques com repositórios falsificados. A URL que eu usava:

```
https://raw.githubusercontent.com/digistump/arduino-boards-
index/master/package_digistump_index.json
```

Deve aparecer a opção de instalar o **Digistump Boards**; é só instalar para carregar os módulos dessa placa. Feito isso, aparece no menu a opção **Digispark Default** — tem que marcar essa board.

Agora é só programar. Vou criar uma chave de teste com 23 caracteres, esta: `%!Pu4-!iA,Xbe+_ayTna1`. Neste ponto você deve colocar o que julga que será a **sua** parte fixa — na prática, os 32 caracteres que a chave vai digitar. O código completo:

```
/*
 * LAB Name: Password Hardware Key
 * Author: Nao Importa WEB
 * For More Info Visit: www.cursohacker.com.br
 */
#include <DigiKeyboard.h>
#define LEDPin    0
#define buttonPin 1
void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(LEDPin, OUTPUT);
}
void loop() {
  boolean buttonState = digitalRead(buttonPin);
  if (buttonState) {
    DigiKeyboard.println("*%!Pu4-!*iA,Xbe+_ayTna1");
```

```
    DigiKeyboard.delay(3000);
  }
  digitalWrite(LEDPin, buttonState);
}
```

Agora você compila o programa e envia para o hardware, que deve estar na USB. Cuidado com extensões de USB: coloque o hardware diretamente na porta do computador. Se tudo der certo, a IDE mostra a gravação concluída.

Usando a chave e o martelo ao lado

Agora você pode usar sua chave. Recomendação: **não decore** esses 32 caracteres. Se um dia for pego, faça como eu — tenho um martelo aqui do lado; uma martelada e o hardware para de funcionar. Como eu não sei o que está gravado ali, ninguém consegue extrair de mim o que eu genuinamente não guardo na cabeça. É a mesma lógica do LUKS destruído: o que não existe mais não pode ser confessado sob pressão.

Para testar, abra o bloco de notas e pressione o botão por um pouco menos de 1 segundo — veja os caracteres serem digitados. O uso real de uma senha importante fica assim:

1. Pressione o botão e veja os asteriscos serem preenchidos (os 32 que eu não sei);
2. Digite os 16 caracteres que você escolheu memorizar (a parte que só existe na sua cabeça).

O resultado é uma senha de 48 caracteres de alta entropia, imune a ser digitada num arquivo, imune a keylogger que dependa de você teclar tudo, e dividida entre algo físico e algo mental. Nenhum dos dois, sozinho, serve para o atacante.

O username também é uma armadilha

Fecho o capítulo voltando ao nome, porque o **username** guarda uma armadilha que já derrubou gente boa. Conheço um hacker até famoso, mas sempre o vejo criar apelidos bem parecidos, todos relacionados a um determinado deus. Aí é fácil: quem observa liga uma persona à outra pelo padrão. Apelidos devem ser complexos? A resposta é simples: depende do objetivo.

Se uma persona é só de zueira, ou realmente precisa ser pública para engajar outros, que seja um apelido simples e humano, pode até repetir algo semelhante. Mas se for uma persona envolvida em ataques e em grupos hacker, que ela seja **sem nexo** com qualquer outra das suas personas públicas. Uma vez quis sacanear outro hacker: fiz uma persona bem parecida com a dele e conduzi alguns ataques a instituições, só para incriminá-lo — afinal, ele sempre criava um nick muito parecido. Se eu consigo fazer isso para incriminar alguém, a polícia e outros hackers também conseguem correlacionar suas personas dessa mesma forma.

Quando desenvolvi um sistema de comunicação e gestão de grupos hacker, projetei uma forma em que é **impossível** um membro dizer como quer ser chamado, e ainda criei um botão que muda o apelido randomicamente, sem nexo com nada. Fiz isso porque muitos hackers são desleixados nesse quesito, e o ego os faz querer sempre ser "aquele hacker conhecido". Lembre-se de que o fim do **LulzSec** foi um hacker desleixado chamado **Hector Xavier Monsegur**, o "Sabu" — preso, ele virou informante do FBI e entregou o grupo por dentro. O rastro de identidade, o ego e a repetição de padrões pesaram tanto quanto qualquer falha técnica.

O problema do username é que a pessoa quer ter uma marca, quer ter um legado — e neste ramo não se deve ter esse legado. Talvez essa ideia tenha nascido da romantização do tema e das grandes histórias, mas na realidade as histórias hacker são histórias tristes. Não queira criar uma marca. Queira criar um futuro seguro para você.

O futuro sem senha: passkeys e FIDO2

Se o password é a origem de tanto sofrimento — reuso, vazamento, força bruta, credential stuffing —, a conclusão natural da indústria foi tentar matar a senha de vez. É o que propõem as **passkeys**, construídas sobre o padrão **FIDO2/WebAuthn**. A ideia liga diretamente ao capítulo de Identidade: em vez de um segredo compartilhado que o servidor guarda (e que pode vazar), usa-se **criptografia de chave pública**. Seu dispositivo gera um par de chaves por serviço; a **chave privada** nunca sai do aparelho (fica protegida por biometria ou PIN, muitas vezes num chip seguro), e o site guarda apenas a **chave pública**. No login, o servidor manda um desafio, seu dispositivo assina com a chave privada, e pronto — não há senha para vazar, não há o que reutilizar entre serviços, e a assinatura é amarrada ao domínio verdadeiro, o que quebra o phishing por definição (a passkey do site falso simplesmente não corresponde).

Do ponto de vista defensivo, é o maior avanço em autenticação em décadas: um banco de dados invadido de um serviço com passkeys não entrega chave privada nenhuma, porque ela nunca esteve lá. Do ponto de vista do hacker, muda o alvo — em vez de caçar a senha, passa-se a caçar o **dispositivo** e o momento em que a chave privada é destravada. É cedo para dizer que a senha morreu, e por muito tempo ainda conviveremos com as duas coisas, mas quem opera precisa entender já para onde a maré vai. A minha chave física de 32+16 caracteres e uma passkey resolvem o mesmo problema por caminhos parecidos: tirar o segredo da sua cabeça e de arquivos vazáveis, e ancorá-lo em algo que você fisicamente possui.

Ferramentas citadas

- **Hydra** (THC-Hydra) — cracker de autenticação online paralelo, para muitos protocolos (SSH, FTP, HTTP, SMB, etc.); **código aberto** (licença AGPLv3, com cláusula própria da THC).
- **John the Ripper** — quebrador de hashes offline (centenas de formatos), tradicionalmente em CPU; **código aberto** (GPLv2, versão community).
- **hashcat** — quebrador de hashes offline acelerado por GPU, o mais usado hoje; **código aberto** (licença MIT).
- **Metasploit Framework** — plataforma de exploração, aqui usada para enumeração de usuários SSH; **código aberto** (licença BSD, edição Framework).
- **Digispark / Arduino (DigiKeyboard)** — microcontrolador que emula teclado USB (HID) para injetar caracteres; **código aberto** (hardware e IDE Arduino sob licenças livres; biblioteca Digistump descontinuada).
- **Have I Been Pwned / Pwned Passwords** — serviço de consulta a credenciais e senhas vazadas; consulta **gratuita**, com API paga e base de dados publicada abertamente.

Referências

- HUNT, Troy. **Have I Been Pwned** e **Pwned Passwords**. Disponível em: https://haveibeenpwned.com/

- NORDPASS. **Top 200 Most Common Passwords**. Relatório anual. Disponível em: https://nordpass.com/most-common-passwords-list/
- OPEN WEB APPLICATION SECURITY PROJECT (OWASP). **Password Storage Cheat Sheet** e **Credential Stuffing Prevention Cheat Sheet**. Disponível em: https://cheatsheetseries.owasp.org/
- FIDO ALLIANCE; W3C. **Web Authentication (WebAuthn)** e **FIDO2/CTAP**. Disponível em: https://fidoalliance.org/
- MUNROE, Randall. **Password Strength**. xkcd nº 936. Disponível em: https://xkcd.com/936/
- GRASSI, Paul A. et al. **NIST Special Publication 800-63B: Digital Identity Guidelines — Authentication and Lifecycle Management**. National Institute of Standards and Technology, 2017.