Comandos curl para testar requisições API

$ curl [OPÇÕES] [URL]

O curl possui dezenas de opções, mas as mais comuns para lidar com APIs são:

Opção	Descrição
-i, --include	Mostra o header da resposta no output
-d, --data	Dados a serem enviados no POST
-H, --header	Envia o header da requisição
-X, --request	Especifica o método HTTP a ser usado na requisição

Teste de Conectividade

Faça uma requisição GET simples para checar se o site responde: curl https://exemplo.com. Para ver só headers: curl -I https://exemplo.com (verifica status como 200 OK ou erros SSL).

Adicione -k para ignorar erros de certificado: curl -k -I https://exemplo.com.

Enumeração de Diretórios

Teste paths comuns: curl -s https://exemplo.com/admin/ | grep -i error (procura erros reveladores como 404 ou 403). Use loops para fuzzing básico: for dir in admin login backup; do curl -s -o /dev/null -w "%{http_code} %{url_effective}\n" https://exemplo.com/$dir/; done.

Testes de Métodos HTTP

Verifique métodos permitidos: curl -X OPTIONS https://exemplo.com -v (mostra Allow: GET,POST).

Teste POST: curl -X POST -d "user=admin&pass=123" https://exemplo.com/login (simula login fraco).

Headers e User-Agent

Spoof User-Agent: curl -A "Mozilla/5.0" https://exemplo.com. Adicione headers custom: curl -H "X-Forwarded-For: 127.0.0.1" https://exemplo.com (testa bypass de IP).

Teste de Vulnerabilidades Básicas

Para SQLi simples: curl "https://exemplo.com/page?id=1' OR '1'='1". Para XSS: curl -d "input=<script>alert(1)</script>" https://exemplo.com/search (observe resposta).

Use -v para verbose e veja tráfego detalhado.

Automação com Script

Crie um bash script para scan rápido:

text

#!/bin/bash

url=$1

curl -s -w "HTTP: %{http_code}, Size: %{size_download}\n" $url/

Execute: ./scan.sh https://exemplo.com.

Limitações e Dicas

Curl é ótimo para recon manual, mas para pentest completo use Burp Suite ou sqlmap. Sempre respeite robots.txt e leis (LGPD no BR). Teste em labs como DVWA primeiro.

Se passando por um user agente

curl -H "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36" -H "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8" -H "Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7" -H "Referer: https://www.google.com/" https://www.google.com

Explorar utilizando o Kali LINUX

1. Reconhecimento e varredura com Kali

1.1 Descoberta de host e serviços (Nmap)

No Kali:

bash

nmap -sV -sC -p 80,443 <IP-ou-dominio>

- -sV: versão de serviços.
- -sC: scripts básicos (inclui alguns checks de vulnerabilidades simples).

Para varrer todas as portas TCP:

bash

nmap -sS -sV -p- <IP>

Use isso em aula para mostrar:

- Identificação de servidor web (Apache, Nginx, IIS).
- Identificação de versão (para discutir CVEs e hardening).

---

2. Mapeamento da aplicação web

2.1 Navegação manual + DevTools

- Usar Firefox/Chromium no Kali, abrir a aplicação.
- Ensinar os alunos a olhar: URLs, parâmetros GET/POST, cookies, cabeçalhos, respostas de erro.

2.2 Descoberta de diretórios (gobuster/dirb)

Exemplo com gobuster:

bash

gobuster dir -u http://<alvo> -w /usr/share/wordlists/dirb/common.txt -x php,html,js

Objetivo em sala:

- Encontrar páginas “escondidas”: /admin, /backup, /test, etc.
- Discutir por que segurança “por obscuridade” não funciona bem.

---

3. Uso de proxy interceptador (Burp Suite)

Burp já vem no Kali. A sequência didática:

1. Abrir Burp Suite (Community).
2. Configurar o navegador para usar o proxy HTTP 127.0.0.1:8080.
3. Habilitar “Intercept” e navegar no site.
4. Mostrar como visualizar e editar requests (parâmetros, cookies, headers).
5. Enviar requests para “Repeater” e “Intruder” para testes mais sistemáticos.

Com isso você ilustra:

- Entendimento do fluxo HTTP/HTTPS.
- Ataques de teste em parâmetros (XSS, SQLi, LFI, etc.).
- Captura de sessões para discutir segurança de autenticação.

---

4. Testes de vulnerabilidades comuns

4.1 SQL Injection com sqlmap

Supondo que você identificou uma URL suspeita (por exemplo, id=1):

bash

sqlmap -u "http://<alvo>/produto.php?id=1" --batch --dbs

bash

sqlmap -u "http://<alvo>/produto.php?id=1" --tables -D <database>

sqlmap -u "http://<alvo>/produto.php?id=1" -T <tabela> -C <colunas> --dump

4.2 XSS com Burp ou manual

- Na própria aplicação vulnerável, inserir payloads simples em campos de input:
    - "><script>alert(1)</script>

- Verificar se o script é refletido na resposta sem sanitização.

Usar Burp:

- Capturar o request, enviar para Repeater.
- Testar várias variações de payloads e contextos (HTML, atributo, JS).

---

5. Scanners de vulnerabilidade web

5.1 Nikto

Ferramenta clássica para servidores web:

bash

nikto -h http://<alvo>

Ela aponta:

- Configurações inseguras.
- Arquivos e scripts perigosos conhecidos.
- Versões com problemas de segurança.

5.2 Outros scanners (para citar)

Você pode citar em aula, mesmo que não use todos:

- OWASP ZAP (proxy + scanner).
- wpscan (para WordPress).
- skipfish, arachni (dependendo da distro/instalação).

Enfatize que scanner não substitui análise manual.

---

6. Relatório

Fazer um relatório com 3 sites de sua escolha, utilizando a seguinte estrutura:

1. Contexto e escopo (alvo, datas, restrições).
2. Metodologia (fases, ferramentas principais, referências OWASP).
3. Achados (para cada vulnerabilidade):
    - Descrição.
    - Evidência (request/response, print de tela).

- Impacto potencial.
- Recomendações (correção, mitigação).

4. Conclusão geral