Glossary & Cheat Sheet

Your complete reference guide for bug bounty hunting fundamentals

Table of Contents

© Wiz Inc. All Rights Reserved.

1 Terminal & Command Line Basics

Navigation Commands

- **`pwd`** – Shows your current directory path
- **`ls`** – Lists files in current directory
- **`ls -la`** – Lists files with detailed information including hidden files
- **`cd folder`** – Changes to specified directory
- **`cd ..`** – Goes up one directory level
- **`mkdir foldername`** – Creates a new directory

File Operations

- **`cat filename`** – Displays entire file content
- **`head filename`** – Shows first 10 lines of a file
- **`tail filename`** – Shows last 10 lines of a file
- **`nano filename`** – Opens file in simple text editor
- **`grep "search" filename`** – Searches for text within files

Essential Shortcuts

- **`Ctrl+C`** – Stops currently running command
- **`Ctrl+Z`** – Suspends current command
- **`Ctrl+R`** – Search command history
- **`Tab`** – Auto-complete file/directory names

tmux (Terminal Multiplexer)

- **`tmux new -s name`** – Create new session with name
- **`Ctrl+b, c`** – Create new window
- **`Ctrl+b, n`** – Switch to next window
- **`Ctrl+b, d`** – Detach from session
- **`tmux attach -t name`** – Reattach to session

nano Text Editor

- **`Ctrl+O`** – Save file (Write Out)
- **`Ctrl+X`** – Exit nano
- **`Ctrl+K`** – Cut (delete) current line
- **`Ctrl+U`** – Paste cut lines
- **`Ctrl+W`** – Search for text

© Wiz Inc. All Rights Reserved.

2 Web Application Fundamentals

Core Concepts

Client-Server Model

- **Client:** Your browser making requests
- **Server:** Web server responding to requests
- **Stateless:** Each HTTP request is independent

DNS (Domain Name System)

- Translates domain names (example.com) into IP addresses
- Essential for finding subdomains during reconnaissance

URL Structure

`https://admin.example.com:8080/api/users?id=123#profile`

- **Protocol:** `https://` (HTTP over encrypted connection)
- **Subdomain:** `admin.` (often indicates different functionality)
- **Root Domain:** `example.com` (main website)
- **Port:** `:8080` (non-standard port, potential admin panels)
- **Path:** `/api/users` (specific endpoint or page)
- **Query Parameters:** `?id=123` (data that can be manipulated)
- **Fragment:** `#profile` (client-side only, not sent to server)

HTTP Basics

HTTP Request Components:

- **Method:** GET (retrieve data), POST (send data), PUT, DELETE
- **Headers:** Metadata like cookies, user-agent, content-type
- **Body:** Data being sent (common with POST requests)

HTTP Request Components:

- **Status Code:** 200 OK, 404 Not Found, 500 Error, etc.
- **Headers:** Instructions for the browser
- **Body:** The actual content (HTML, JSON, etc.)

© Wiz Inc. All Rights Reserved.

Key HTTP Status Codes

Code	Meaning	Bug Hunter Focus
200 OK	Success	Did an unauthorized action succeed?
401 Unauthorized	Login Required	Can this be bypassed?
403 Forbidden	Access Denied	Can this restriction be bypassed?
404 Not Found	Doesn't Exist	Does the error message leak info?
500 Server Error	Server Problem	Does the error expose stack traces?

Cookies and Sessions

Cookie: Small piece of data stored by browser, sent with every request

Session: Server-side record of user state, identified by session ID in cookie

Cookie Security Flags:

- **HttpOnly:** Prevents JavaScript access (stops XSS attacks)
- **Secure:** Only sends cookie over HTTPS
- **SameSite:** Prevents cross-site request forgery

Frontend vs Backend

- **Frontend (Client-Side):** What users see - HTML, CSS, JavaScript in browser
- **Backend (Server-Side):** Hidden server logic, databases, authentication
- **Key Principle:** Never trust frontend restrictions - always test backend enforcement

APIs (Application Programming Interfaces)

- Endpoints that return raw data (usually JSON) instead of HTML pages
- Often less protected than main web interface
- Direct testing bypasses frontend limitations

© Wiz Inc. All Rights Reserved.

3 Common Vulnerability Types

IDOR (Insecure Direct Object Reference)

What it is: Change a parameter to access other users' data

Example: `/api/profile?user_id=123` → change to `user_id=124`

Why it happens: Server doesn't check if you own the requested resource

Testing: Look for numeric IDs in URLs and API calls, try different values

SSRF (Server-Side Request Forgery)

What it is: Trick server into making requests to internal systems

Example: Image upload that accepts URLs - provide internal server address

Common targets:

- `http://169.254.169.254/` (AWS metadata)
- `http://localhost:8080/admin`
- Internal network ranges (10.x.x.x, 192.168.x.x)

Subdomain Takeover

What it is: Control company subdomain by claiming underlying service

Example: blog.company.com points to unclaimed GitHub Pages site

How to find: Use tools like subfinder and check for unclaimed services

Impact: Can serve content as if it came from the company

Directory & File Discovery

What it is: Find hidden directories, backup files, or admin panels

Common targets:

- `/admin/`,`/backup/`,`/staging/`
- Files with extensions like `.bak`,`.old`,`.backup`
- `/.git/` (exposed source code)
- `/uploads/` with directory listing

© Wiz Inc. All Rights Reserved.

GitHub Secrets & Information Disclosure

What it is: Sensitive information in public repositories

What to look for:

- API keys and tokens
- Internal domain names and URLs
- Database credentials
- Configuration files with sensitive data

Authentication & Session Management Flaws

Common issues:

- Session cookies working across environments (staging → production)
- Predictable session tokens
- Sessions not expiring properly
- Missing authentication on sensitive endpoints

Business Logic Vulnerabilities

What it is: Application works as designed, but logic is flawed

Example:

- Race conditions in payment processing
- Referral systems that can be gamed
- Multi-factor authentication bypasses
- Price manipulation through parameter tampering

CVEs (Common Vulnerabilities and Exposures)

What it is: Publicly disclosed vulnerabilities in popular software

Opportunity: Test new CVEs against targets before they patch

Approach: Monitor CVE feeds, set up alerts, test quickly

4 Essential Tools Reference

Web Proxy Tools

Burp Suite Community (Free + Paid)

- Download: portswigger.net/burp/communitydownload
- Purpose: Intercept and modify HTTP requests
- Essential for manual web application testing

© Wiz Inc. All Rights Reserved.

Caido (Free + Paid)

- Download: https://caido.io/download
- Modern interface with sleek UX written in Rust
- Purpose: Intercept and modify HTTP requests

Reconnaissance Tools

Installation Commands:

```
# Subdomain discovery
go install github.com/projectdiscovery/subfinder/v2/cmd/subfinder@latest
# HTTP probing
go install github.com/projectdiscovery/httpx/cmd/httpx@latest
# Vulnerability scanning
go install github.com/projectdiscovery/nuclei/v2/cmd/nuclei@latest
# Port scanning
go install github.com/projectdiscovery/naabu/v2/cmd/naabu@latest
# DNS resolution
go install github.com/d3mondev/puredns/v2@latest
```

Tool Purposes:

- **subfinder:** Discovers subdomains using multiple sources
- **httpx:** Checks which hosts are alive and responds with HTTP
- **nuclei:** Scans for known vulnerabilities using templates
- **naabu:** Fast port scanning
- **puredns:** DNS resolution and validation

Fuzzing & Directory Discovery

```
# Web fuzzing
go install github.com/ffuf/ffuf@latest
# Alternative directory discovery
go install github.com/OJ/gobuster/v3@latest
```

© Wiz Inc. All Rights Reserved.

Common Usage:

```
# Directory fuzzing
ffuf -w /path/to/wordlist -u https://target.com/FUZZ
# Subdomain fuzzing
ffuf -w /path/to/wordlist -u https://FUZZ.target.com
```

Wordlists

```
# Download SecLists (comprehensive wordlists)
git clone https://github.com/danielmiessler/SecLists.git ~/wordlists
```

Common wordlist locations:

```
~/wordlists/SecLists/Discovery/Web-Content/common.txt
~/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
```

Supporting Tools

```
# System utilities
sudo apt install jq curl wget git
# JSON processing
jq
# Network testing
curl -X GET https://example.com
wget https://example.com/file.txt
```

© Wiz Inc. All Rights Reserved.

5 Bug Bounty Program Terms

Core Terminology

Bounty: Reward (money, swag, or recognition) for finding valid bugs

Hunter/Researcher: Person testing systems and finding vulnerabilities

Program: Company/platform inviting ethical hackers to test their systems

Scope: Specific assets (websites, APIs, apps) that are legal to test

Out of Scope: Assets you're NOT allowed to test (can result in program ban)

PoC (Proof of Concept): Clear demonstration showing how to reproduce a bug

Triage: Team/process that validates bug reports before payout

Severity Classifications

Critical: Complete system compromise, mass data exposure

- Typical payout: $5,000 - $100,000+

High: Significant data access, privilege escalation

- Typical payout: $1,000 - $15,000

Medium: Limited data exposure, some functionality bypass

- Typical payout: $200 - $2,000

Low: Information disclosure, minor functionality issues

- Typical payout: $50 - $500

Program Types

VDP (Vulnerability Disclosure Program): Recognition/thanks only, no money

BBP (Bug Bounty Program): Monetary rewards for valid findings

Private Program: Invitation-only, usually higher payouts

Public Program: Open to all researchers

Report Status Types

New: Just submitted, awaiting triage

Triaged: Validated by triage team, sent to company

Resolved: Company has fixed the issue

Duplicate: Same bug already reported by someone else

N/A (Not Applicable): Not a valid security issue

Informative: Valid observation but low/no security impact

© Wiz Inc. All Rights Reserved.

6 Community & Collaboration

Why Collaboration Matters

- Different hunters excel at different skills (automation, manual testing, reporting)
- Faster learning through knowledge sharing
- Better results when combining complementary skills
- More fun and motivation

Ways to Collaborate

Live Hacking Events:

- Teams often outperform individual hunters
- Typical split: one person does recon, another does manual testing, another handles reporting

Online Communities:

- Discord/Slack servers for real-time help
- X (Formerly Twitter) for following discoveries and joining conversations

Knowledge Sharing:

- Write blog posts about your findings
- Share tools and methodologies
- Contribute to open source security tools
- Speak at conferences and meetups

Community Etiquette

- Credit others' work when building on it
- Don't share active targets or live vulnerabilities
- Help others without expecting immediate returns
- Accept feedback gracefully
- Be respectful and professional

Getting Started in Community

1. Join 2-3 bug bounty Discord servers
2. Follow 10-15 active hunters on X (Formerly Twitter)
3. Introduce yourself in newcomer channels
4. Ask thoughtful questions about what you're learning
5. Share resources that helped you recently

© Wiz Inc. All Rights Reserved.

7 Quick Command Reference

Basic Reconnaissance Workflow

```
# Create organized workspace
mkdir -p ~/bugbounty/target.com/{recon,scans,notes}
cd ~/bugbounty/target.com/recon
# Subdomain discovery
subfinder -d target.com -o subdomains.txt
# Check which subdomains are alive
cat subdomains.txt | httpx -status-code -o live_hosts.txt
# Vulnerability scanning
cat live_hosts.txt | nuclei -t ~/nuclei-templates/ -o vulnerabilities.txt
# Directory fuzzing on interesting targets
ffuf -w ~/wordlists/SecLists/Discovery/Web-Content/common.txt -u https://
target.com/FUZZ
```

Tool Chaining Examples

```
# Subdomain discovery → live checking → vulnerability scanning
subfinder -d target.com | httpx | nuclei -t vulnerabilities/
# Find subdomains → save results while processing
subfinder -d target.com | tee subdomains.txt | httpx | tee live_hosts.txt
# Port scanning on discovered hosts
cat live_hosts.txt | naabu -top-ports 100
```

Data Processing

```
# Remove duplicates from results
sort results.txt | uniq > unique_results.txt
# Count lines in file
wc -l filename.txt
# Search for specific content
grep "admin" results.txt
# Filter out specific content
grep -v "unwanted" results.txt > filtered.txt
```

© Wiz Inc. All Rights Reserved.

File Organization

```
# Typical bug bounty directory structure
~/bugbounty/
├── target1.com/
│ ├── recon/
│ ├── scans/
│ └── notes/
├── target2.com/
│ ├── recon/
│ ├── scans/
│ └── notes/
├── tools/
└── wordlists/
```

8 Getting Help & Troubleshooting

Common Issues

"Command not found" errors:

- Check if tool is installed: `which toolname`
- Verify PATH includes Go bin: `echo $PATH`
- Reinstall tool: `go install [tool-url]@latest`

Permission denied errors:

- Make scripts executable: `chmod +x script.sh`
- Check file ownership: `ls -la filename`

Tool installation fails:

- Update Go: `go version` (should be 1.19+)
- Clear Go cache: `go clean -cache`
- Check internet connection

Where to Get Help

Communities:

- Bug bounty Discord servers
- X (Formerly Twitter) bug bounty community

© Wiz Inc. All Rights Reserved.

Documentation:

- Tool GitHub repositories
- Official documentation sites
- Community-written tutorials

When Asking for Help:

- Describe what you're trying to accomplish
- Show what you've already tried
- Include specific error messages
- Provide relevant screenshots or logs

9 Final Tips for Success

Mindset

- Focus on understanding concepts, not memorizing details
- It's okay to look things up - even experts use references
- Start with basics and gradually build complexity
- Be patient - bug bounty skills develop over time

Continuous Learning

- Follow security researchers on X (Formerly Twitter)
- Read bug bounty write-ups and case studies
- Practice on legal targets and lab environments
- Join the community and ask questions

Professional Development

- Document your learning journey
- Build a portfolio of findings and write-ups
- Network with other security professionals
- Consider how bug bounty skills apply to security careers

This reference guide covers the foundational knowledge from the Wiz Bug Bounty Course. Keep it handy during your bug hunting journey and don't hesitate to add your own notes and discoveries.

© Wiz Inc. All Rights Reserved.

In Summary

The foundations section covered a lot of ground - terminal skills, web application basics, common vulnerabilities, essential tools, and community aspects.

Rather than expecting you to memorize everything, we've compiled all the key terms, concepts, and commands into a comprehensive reference guide.

Download Your Cheat Sheet

What's included:

- All terminal commands from the bash primer
- Web application terminology and concepts
- Complete vulnerability type definitions
- Tool installation commands and basic usage
- Bug bounty program terminology
- Community and collaboration terms

Format: Clean, searchable PDF optimized for quick reference during actual bug hunting

How Professional Hunters Use References

Professional bug bounty hunters don't memorize everything. They:

- Keep command references handy for complex tool syntax
- Reference vulnerability definitions when writing reports
- Look up HTTP status codes and headers regularly
- Check terminology when communicating with other hunters

The key: Focus on understanding concepts, not memorizing details.

Using This Guide Effectively

During Active Hunting

- Keep the PDF open on a second monitor or device
- Quick searches for forgotten command syntax
- Double-check vulnerability classifications before reporting
- Reference proper terminology for professional reports

© Wiz Inc. All Rights Reserved.

While Learning

- Annotate the PDF with your own notes and discoveries
- Highlight sections relevant to your preferred hunting style
- Add bookmarks for frequently referenced sections
- Use it to review concepts before practice sessions

Building Your Knowledge

As you gain experience:

- Add your own tool discoveries and command variations
- Include notes about what works best for different target types
- Build a personal methodology section
- Share useful additions with the community

What's Not Included (And That's OK)

This reference covers the **foundations** – the core knowledge you need to get started and understand the rest of the course.

Advanced topics like specific exploitation techniques, complex tool configurations, and detailed reporting templates will be covered in their respective course sections.

The philosophy: Master the basics first, then build advanced skills on that foundation.

Key Takeaway

Bug bounty success isn't about having perfect memory – it's about:

- Understanding core concepts deeply
- Knowing where to find details when you need them
- Having reliable references during active hunting
- Continuously building your knowledge base

Download your cheat sheet now and use it as your constant companion throughout the rest of the course and beyond.

© Wiz Inc. All Rights Reserved.