



OFFENSIVE SECURITY REFERENCE

Bug Bounty & Web Security. The Complete Hunter's Guide

A structured, end-to-end reference for ethical hackers — methodology, reconnaissance, the OWASP Top 10, common vulnerability classes, tooling, report writing, and the ethics that keep it all legal.

Methodology

Reconnaissance

OWASP Top 10

Vulnerability Classes

Tooling

Reporting & Ethics

Career Growth

SECTIONS

9

VULN CLASSES

20+

EDITION

2026

Table of Contents

01 Introduction & Fundamentals	What bug bounty is, skills, mindset, legality
02 Platforms & Program Types	HackerOne, Bugcrowd, VDPs, private programs
03 Methodology & Hunting Process	The end-to-end workflow of a hunt
04 Reconnaissance & Enumeration	Mapping the attack surface
05 OWASP Top 10 (2021)	The industry-standard risk categories
06 Common Vulnerability Classes	The bugs hunters find most often
07 Essential Tools & Tech Stack	Burp Suite, recon tools, scanners
08 Reporting, Ethics & Disclosure	Writing reports that get triaged and paid
09 Career Growth & Resources	Labs, certifications, community

01

Introduction & Fundamentals.

Bug bounty hunting is authorized, compensated vulnerability research: finding security flaws in a company's systems within a defined scope and reporting them responsibly, instead of exploiting them.

[What is Bug Bounty](#)[vs. Pentesting](#)[Core Skills](#)[Legal Baseline](#)[Hunter Mindset](#)

01 Introduction & Fundamentals

Everything here assumes work performed strictly within an authorized program's published scope. Testing systems without permission is illegal in most jurisdictions, regardless of intent.

What Is Bug Bounty Hunting

A crowdsourced security model where organizations invite independent researchers to find vulnerabilities in their applications and infrastructure, paying a bounty for each valid, in-scope finding. It complements — rather than replaces — internal security teams and formal audits.

Bug Bounty vs. Penetration Testing

Penetration testing is a time-boxed, contracted engagement with a fixed scope and a single tester or small team, delivering a comprehensive report by a deadline. Bug bounty is continuous and crowdsourced — hundreds of researchers probe the same scope indefinitely, and payment is per-finding rather than per-hour.

Core Skills Foundation

Solid grounding in how the web works: HTTP/HTTPS request-response cycles, cookies and sessions, REST/GraphQL APIs, the DOM, and common server-side stacks. Comfort with Linux command-line tools, at least one scripting language (Python or JavaScript), and basic networking (DNS, TCP/IP) round out the baseline.

Legal & Ethical Baseline

Only test assets explicitly listed in a program's scope, using only the techniques it permits (many prohibit automated scanning, social engineering, or physical access). Most programs publish a "safe harbor" clause protecting good-faith researchers from legal action — read it before you start, not after you find something.

The Hunter's Mindset

Successful hunters are systematic rather than random: they take detailed notes, revisit assumptions, read source code and documentation closely, and treat "boring" features (password reset, file upload, account settings) as seriously as flashy ones — because that's usually where the bugs live.

02

Platforms & Program Types.

Most bug bounty activity flows through a handful of managed platforms that handle scope publication, triage, payments, and researcher reputation — rather than researchers negotiating directly with every company.

- HackerOne
- Bugcrowd
- Intigriti
- YesWeHack
- Synack
- VDPs

02 Platforms & Program Types

Each platform has its own triage speed, payout norms, and community — many hunters focus on two or three rather than spreading thin across all of them.

HackerOne

PLATFORM

The largest bug bounty platform by program count, hosting both public and private programs for companies like GitHub, Shopify, and the U.S. Department of Defense. Known for its reputation/signal system and live-hacking events.

Bugcrowd

PLATFORM

A major competitor to HackerOne with a strong managed-triage service, a points-based researcher ranking system, and its own set of well-regarded public programs.

Intigriti

PLATFORM

Europe-based platform popular for programs from European enterprises, with a reputation for fast triage response times.

YesWeHack

PLATFORM

French-founded platform with a growing catalog of European and public-sector programs, plus a dojo-style learning portal.

Synack

INVITE-ONLY

A vetted, invite-only researcher pool ("Synack Red Team") combined with automated scanning, typically offering higher and more consistent payouts in exchange for a selective application process.

Vulnerability Disclosure Programs (VDPs)

NO BOUNTY

Programs that accept and fix vulnerability reports but do not pay cash — often the only legal way to test a given company, and a good way to build reputation and a track record.

Program Scope Documents

Every program publishes in-scope and out-of-scope assets, accepted vulnerability types, prohibited techniques, and reward ranges. Re-reading the scope before every session — not just once — is one of the most common ways hunters avoid wasted or invalid reports.

03

Methodology & Hunting Process.

Experienced hunters follow a repeatable process rather than clicking around randomly — it's what turns a large, unfamiliar target into a manageable, prioritized list of things to test.

- Scope Review
- Reconnaissance
- Mapping
- Vulnerability Analysis
- Exploitation & Verification
- Reporting
- Retesting

03 Methodology & Hunting Process

A typical hunt moves through these phases in order, though experienced hunters loop back to earlier phases constantly as new information surfaces.

1. Scope Review

Read the program's policy fully: in-scope domains, excluded assets, disallowed techniques, and known/duplicate issues. Decisions made here prevent wasted effort and invalid submissions later.

2. Reconnaissance

Passively and actively map the target's attack surface — subdomains, exposed services, technology stack, and publicly accessible code or documentation — before touching any specific feature.

3. Mapping & Enumeration

Walk through the application as a normal user, then as an authenticated user across every role available, cataloging every endpoint, parameter, and workflow — this becomes your personal checklist for the next phase.

4. Vulnerability Analysis

Systematically test each mapped endpoint against relevant vulnerability classes (see Sections 05–06), starting with the areas most likely to contain business logic or access-control flaws.

5. Exploitation & Impact Verification

Confirm a suspected issue is real and demonstrate its actual impact with the minimum necessary proof — enough to convince a triager, without going further than needed to prove the point (e.g., reading one record, not dumping a database).

6. Reporting

Write up the finding clearly and submit it through the platform (see Section 08 for structure). Speed matters — bug bounty is competitive, and duplicate reports are typically not paid.

7. Retesting & Follow-Up

Once a fix is deployed, confirm it actually resolves the issue when asked to retest, and stay responsive to triager questions — reports that go quiet are more likely to be closed as unresponsive.

04

Reconnaissance & Enumeration.

Good recon turns an intimidating target into a concrete list of hosts, endpoints, and technologies — most high-value findings come from areas competitors overlooked during this phase.

Subdomain Enumeration

Content Discovery

Tech Fingerprinting

OSINT

JS File Analysis

Cloud Asset Discovery

04 Reconnaissance & Enumeration

The goal of recon is coverage: finding every in-scope asset a target owns, not just the main application everyone else is already testing.

Subdomain Enumeration

Passive tools (crt.sh certificate transparency logs, Subfinder, Amass) and active DNS brute-forcing reveal forgotten staging, admin, or legacy subdomains that often carry weaker security controls than the flagship site.

Content Discovery

Directory and file brute-forcing (ffuf, dirsearch, Gobuster) against discovered hosts, combined with the Wayback Machine and GAU to surface old, unlinked, or removed endpoints that may still be live.

Technology Fingerprinting

Identifying the frameworks, CMS, server software, and third-party libraries in use (via Wappalyzer, response headers, or error messages) to prioritize testing toward known weaknesses in that specific stack.

OSINT & Google Dorking

Search operators (site:, filetype:, intitle:) and public code repositories can surface exposed configuration files, API keys, internal documentation, or employee information relevant to social engineering-adjacent risk (where in scope).

JavaScript File Analysis

Client-side JS bundles frequently embed hard-coded API endpoints, internal hostnames, feature flags, and occasionally credentials or tokens — worth systematically reviewing rather than skipping.

API Endpoint Discovery

Reviewing Swagger/OpenAPI docs, Postman collections, and mobile app traffic (via a proxy) to find API routes not exposed in the web UI.

Cloud Asset & Bucket Discovery

Enumerating cloud storage buckets and services tied to the target's naming conventions, checking for misconfigured public access — always within program scope.

05

OWASP Top 10.

The OWASP Top 10 is the industry-standard awareness document ranking the most critical web application security risks, used as a shared vocabulary between hunters, developers, and program owners.

A01 Broken Access Control

A02 Cryptographic Failures

A03 Injection

A04 Insecure Design

A05 Security Misconfiguration

A06 Vulnerable Components

A07 Auth Failures

A08 Integrity Failures

A09 Logging Failures

A10 SSRF

05 OWASP Top 10 (2021) PART 1 OF 2

The current OWASP Top 10 (2021 edition) ranks web application risk categories from real-world vulnerability data. Categories A01–A05 below.

A01 — Broken Access Control

#1 RISK

Users acting outside their intended permissions — viewing or modifying another user's data, accessing admin functionality, or bypassing authorization checks. The single most reported category in the 2021 update, and consistently one of the highest-paying bug classes in bounty programs.

A02 — Cryptographic Failures

DATA EXPOSURE

Sensitive data exposed due to weak, missing, or misused encryption — transmitting credentials over plain HTTP, storing passwords with outdated hashing, or using predictable tokens for password reset links.

A03 — Injection

CLASSIC RISK

Untrusted input interpreted as code or commands by an interpreter — SQL injection, command injection, and NoSQL injection are the most common variants, arising whenever user input reaches a query or system call without proper parameterization.

A04 — Insecure Design

ARCHITECTURAL

Flaws baked into an application's design rather than its implementation — missing rate limiting on sensitive actions, trusting client-side validation alone, or workflows that assume good-faith users by default.

A05 — Security Misconfiguration

CONFIGURATION

Insecure default settings, unnecessary features enabled, verbose error messages leaking stack traces, or missing security headers — often the fastest category to find and confirm, and a frequent starting point for hunters.

05 OWASP Top 10 (2021) PART 2 OF 2

Categories A06–A10, continued from the previous page.

A06 — Vulnerable & Outdated Components

SUPPLY CHAIN

Using libraries, frameworks, or dependencies with known public vulnerabilities. Attackers routinely scan for identifiable version numbers of outdated software and match them against public CVE databases.

A07 — Identification & Authentication Failures

AUTHN

Weaknesses in login, session management, or credential handling — missing multi-factor authentication, predictable session tokens, or account enumeration through inconsistent error messages.

A08 — Software & Data Integrity Failures

INTEGRITY

Code and infrastructure that don't verify integrity before trusting them — insecure deserialization, unsigned auto-update mechanisms, or CI/CD pipelines that pull unverified third-party code.

A09 — Security Logging & Monitoring Failures

DETECTION

Insufficient logging, alerting, or monitoring that lets breaches go undetected for extended periods, delaying incident response even when other controls eventually fail.

A10 — Server-Side Request Forgery (SSRF)

NEWEST ENTRY

An application fetches a remote resource using a URL supplied (directly or indirectly) by the user, without validating the destination — letting an attacker reach internal services, cloud metadata endpoints, or otherwise unreachable systems.

Note: The OWASP Top 10 is a risk-awareness list, not a complete testing checklist. Section 06 covers additional vulnerability classes that bug bounty hunters encounter constantly but that don't map to a single OWASP category.

06

Common Vulnerability Classes.

Beyond the OWASP Top 10's broad categories, these specific bug classes account for the majority of valid, paid bug bounty reports across most programs.

- XSS
- IDOR / BOLA
- CSRF
- SSRF Deep Dive
- XXE
- Subdomain Takeover
- Open Redirect
- Race Conditions
- File Upload Flaws
- Business Logic
- JWT / Session Flaws

06 Common Vulnerability Classes PART 1 OF 2

Conceptual overviews of the bug classes hunters test for most — described at the level needed to recognize and report them, not as attack tooling.

Cross-Site Scripting (XSS)

REFLECTED / STORED / DOM

Occurs when user-controlled input is rendered in a page without proper output encoding, letting an attacker run script in another user's browser. Reflected XSS returns input immediately in the response; Stored XSS persists it in a database for later victims; DOM-based XSS happens entirely in client-side JavaScript without a server round-trip.

IDOR / Broken Object Level Authorization

ACCESS CONTROL

An application exposes a direct reference to an internal object (a record ID, file name, or key) and fails to verify the requesting user is actually authorized to access that specific object — one of the highest-frequency, highest-payout classes in modern bug bounty.

Cross-Site Request Forgery (CSRF)

SESSION RIDING

Tricks an authenticated user's browser into submitting an unwanted request to an application, exploiting the fact that cookies are attached automatically. Modern frameworks mitigate this with anti-CSRF tokens and SameSite cookie attributes, but gaps remain common on legacy or custom endpoints.

SSRF — Practical Impact

A10 DEEP DIVE

Beyond the OWASP definition, SSRF is especially dangerous in cloud environments where it can reach instance metadata services to extract temporary cloud credentials — frequently escalating a "medium" finding into a critical one.

XML External Entity (XXE) Injection

PARSER FLAW

Occurs when a poorly configured XML parser processes external entity references, potentially allowing an attacker to read local files, perform SSRF, or cause denial of service — most common in legacy SOAP APIs and document-processing features.

Subdomain Takeover

DNS MISCONFIG

A DNS record points to a third-party service (a cloud host, SaaS app) that's no longer claimed, letting an attacker register that service and serve content under the victim's domain.

Open Redirect

LOW-MEDIUM

An application redirects users to a URL supplied via a parameter without validating it stays in-domain, enabling phishing links that appear to originate from a trusted site.

06 Common Vulnerability Classes PART 2 OF 2

Continued: additional classes especially relevant to modern APIs and application logic.

Race Conditions

TIMING FLAW

Occurs when an application's logic assumes operations happen sequentially, but concurrent requests reach a shared resource at nearly the same time — used to bypass one-time-use limits, redeem a discount code multiple times, or double-spend a balance.

File Upload Vulnerabilities

INPUT HANDLING

Insufficient validation of uploaded file type, content, or storage location, which can lead to stored XSS (via an uploaded HTML/SVG file), path traversal, or in severe cases remote code execution if executable files reach a web-accessible directory.

Business Logic Flaws

DESIGN-LEVEL

Bugs in the intended workflow rather than the code's syntax — skipping a required payment step, manipulating quantities to produce negative prices, or abusing multi-step processes out of their expected order. These require understanding the product, not just the code, and are rarely caught by automated scanners.

JWT & Session Management Flaws

AUTHN/AUTHZ

Common issues include accepting tokens signed with the "none" algorithm, failing to verify signatures at all, weak or leaked signing secrets, and sessions that don't expire or invalidate properly on logout or password change.

Mass Assignment / Excessive Data Exposure

API SECURITY

APIs that bind incoming request fields directly to internal data models can let an attacker set fields they shouldn't control (like an "isAdmin" flag), while over-permissive responses can leak internal fields never meant to reach the client.

Rate Limiting & Brute Force Gaps

API SECURITY

Missing or weak rate limiting on login, OTP verification, or password reset endpoints allows credential stuffing, OTP brute-forcing, or account enumeration at scale.

07

Essential Tools & Tech Stack.

A relatively small toolkit, used well and repeatedly, outperforms a large collection of tools used superficially. Most experienced hunters build their workflow around one core proxy plus a handful of recon utilities.

- Burp Suite

OWASP ZAP

Nmap

Subfinder / Amass

ffuf / Gobuster

Nuclei
- Postman

Browser DevTools

07 Essential Tools & Tech Stack

These are standard, publicly documented security tools used for authorized testing under a defined scope.

Burp Suite

CORE PROXY

The industry-standard intercepting proxy for manual web testing — captures, inspects, and modifies HTTP traffic in real time. The Community edition covers manual testing; the Professional edition adds an active scanner and advanced automation via extensions.

OWASP ZAP

FREE PROXY

A free, open-source alternative to Burp maintained by OWASP itself, offering automated scanning, a scripting console, and a lower barrier to entry for newcomers.

Nmap

NETWORK SCANNING

The standard tool for network discovery and port scanning, used to identify open ports, running services, and their versions on in-scope hosts.

Subfinder & Amass

RECON

Passive subdomain enumeration tools that query multiple public data sources to build a comprehensive list of an organization's subdomains.

ffuf, Gobuster & dirsearch

CONTENT DISCOVERY

Fast, wordlist-driven fuzzers for discovering hidden directories, files, and parameters on a web server.

Nuclei

TEMPLATE SCANNER

A fast vulnerability scanner driven by community-maintained YAML templates, useful for quickly checking known misconfigurations and CVEs across many hosts within scope.

Postman / Insomnia

API TESTING

GUI tools for constructing, saving, and replaying API requests — essential for systematically testing REST and GraphQL endpoints.

Browser DevTools

CLIENT-SIDE

Built into every modern browser — indispensable for inspecting the DOM, network requests, local storage, and client-side JavaScript logic directly.

08

Reporting, Ethics & Disclosure.

A well-written report is the difference between a bounty and a "not applicable" — clarity and reproducibility matter as much as the finding itself.

[Report Structure](#)[Proof of Concept](#)[CVSS Scoring](#)[Responsible Disclosure](#)[Triage Etiquette](#)[Common Mistakes](#)

08 Reporting, Ethics & Disclosure

Triagers read hundreds of reports — the clearest, most reproducible ones get resolved and paid fastest.

Anatomy of a Strong Report

A clear, descriptive title; a concise summary of the vulnerability and its business impact; exact, numbered steps to reproduce; a minimal proof of concept (screenshot, video, or request/response pair); and a suggested remediation. Vague titles like "XSS found" get deprioritized against specific ones.

Proof of Concept Discipline

Demonstrate impact with the minimum necessary action — reading one non-sensitive record to prove an IDOR, not exporting an entire database. Excessive or destructive proof-of-concept actions can violate program rules even when the underlying bug is valid.

CVSS Scoring Basics

The Common Vulnerability Scoring System rates severity from 0–10 based on factors like attack vector, complexity, privileges required, and impact on confidentiality, integrity, and availability — many programs ask researchers to propose a CVSS vector alongside a report.

Responsible / Coordinated Disclosure

Reports go directly to the program, never publicly, until the organization has resolved the issue and given explicit permission to disclose. Most programs specify a disclosure timeline (often 90 days) after which public write-ups become permitted.

Working With Triage Teams

Respond promptly to clarifying questions, avoid re-submitting the same finding repeatedly, and accept "not applicable" or "duplicate" resolutions professionally — escalation options exist on most platforms for genuine disagreements, but they're for exceptions, not routine disputes.

Common Report-Killing Mistakes

Testing out-of-scope assets, submitting theoretical issues with no real-world impact, missing basic reproduction steps, and reporting already-known/duplicate issues account for the majority of rejected submissions.

09

Career Growth & Resources.

Bug bounty skill compounds through deliberate practice — structured labs, CTFs, and an active community accelerate the learning curve far more than unguided target-hopping.

- PortSwigger Academy
- HackTheBox / TryHackMe
- CTFs
- Certifications
- Community
- Write-ups

09 Career Growth & Resources

Consistent, structured practice beats sporadic effort — most successful hunters treat the first six months as a deliberate training period before expecting significant payouts.

PortSwigger Web Security Academy

FREE TRAINING

A free, hands-on curriculum covering nearly every vulnerability class in this guide with interactive labs, built and maintained by the makers of Burp Suite — widely considered the best free starting point for web security fundamentals.

HackTheBox & TryHackMe

PRACTICE LABS

Gamified platforms with vulnerable machines and guided rooms spanning web, network, and general penetration testing skills, useful for building broader offensive security fundamentals beyond pure web bugs.

Capture The Flag (CTF) Competitions

SKILL BUILDING

Timed competitions with security puzzles across categories like web, crypto, and reverse engineering — a low-stakes way to practice under time pressure and learn from published post-competition write-ups.

Relevant Certifications

CREDENTIALS

OSCP and OSWE (offensive, hands-on), eWPT/eWPTX (web-focused), and CEH (broad foundational) are commonly referenced, though most bug bounty hunters build reputation through platform track record rather than certificates alone.

Community & Events

NETWORK

Infosec Twitter/X, Discord servers tied to specific platforms, and live-hacking events or conferences (DEF CON, Nullcon, local BSides chapters) are where a lot of practical, non-published knowledge circulates.

Publishing Write-Ups

REPUTATION

Once a disclosure window closes and a program permits publication, writing up interesting findings builds public credibility, helps other hunters learn, and often leads to private program invitations.

Hunt within the rules.

Every technique in this guide is meant for use against systems you are explicitly authorized to test — through a published bug bounty program, a vulnerability disclosure program, or a signed penetration testing agreement. Testing systems without authorization is illegal in most jurisdictions, regardless of intent or outcome.

Vulnerability classes and tooling evolve constantly — new OWASP revisions, platform policy changes, and emerging bug classes (particularly around AI/LLM integrations and modern API architectures) mean continuous learning is part of the discipline, not a one-time setup cost.

OWASP Top 10	owasp.org/Top10
Web Security Academy	portswigger.net/web-security
HackerOne Hacktivity	hackerone.com/hacktivity
Bugcrowd University	bugcrowd.com/hackers/bugcrowd-university