🆓 Bug Bounty Checklist & Recon Template

> *The complete free resource mentioned in the video - everything you need to structure your workflow and turn chaos into strategy*

🎯 Pre-Hunt Preparation Checklist

✅ Program Research

- [ ] Read program rules completely (twice!)
- [ ] Identify in-scope assets and domains
- [ ] Note out-of-scope restrictions
- [ ] Check forbidden endpoints/actions
- [ ] Understand acceptable testing methods
- [ ] Review past disclosed reports for patterns
- [ ] Join program's communication channels
- [ ] Set up proper testing environment

✅ Tool Preparation

- [ ] Burp Suite configured and ready
- [ ] Custom wordlists prepared
- [ ] VPN/proxy setup verified
- [ ] Screenshot tools ready
- [ ] Note-taking system established
- [ ] Backup documentation method ready

🔍 Systematic Recon Template

Phase 1: Information Gathering 🕵️‍♀️

Subdomain Discovery

- [ ] Certificate transparency logs (crt.sh)
- [ ] DNS brute forcing with quality wordlists
- [ ] Search engine dorking (`site:target.com`)
- [ ] GitHub/GitLab repository searches
- [ ] Social media and public documents
- [ ] Third-party service integrations

- [ ] Historical DNS data (SecurityTrails, etc.)

Documentation Template:

```
Target: [TARGET_NAME]
Date: [DATE]

Subdomains Found:
- subdomain1.target.com [STATUS_CODE]
- subdomain2.target.com [STATUS_CODE]

Interesting Findings:
-
```

Port & Service Enumeration

- [ ] Nmap comprehensive scan
- [ ] Service version identification
- [ ] Banner grabbing
- [ ] SSL/TLS configuration review
- [ ] Uncommon port discovery
- [ ] Service-specific vulnerability checks

Documentation Template:

```
Port Scan Results:
- Port 80: HTTP [Server Info]
- Port 443: HTTPS [SSL Details]
- Port 8080: HTTP [Additional Service]

Service Versions:
-

Security Observations:
-
```

Phase 2: Web Application Analysis 🌐

Technology Stack Identification

- [ ] HTTP response headers analysis
- [ ] JavaScript framework detection
- [ ] CMS/platform identification (Wappalyzer)
- [ ] Third-party service integration mapping
- [ ] CDN and hosting provider identification
- [ ] Database technology indicators

Technology Stack Template:

```
Web Server: [Apache/Nginx/IIS]
Framework: [React/Angular/Vue/PHP/etc.]
CMS: [WordPress/Drupal/Custom]
Database: [MySQL/PostgreSQL/MongoDB]
CDN: [Cloudflare/AWS/etc.]
Security: [WAF detected/Headers present]

Interesting Technologies:
-
```

Content Discovery

- [ ] Directory brute forcing (common paths)
- [ ] File extension discovery
- [ ] Backup file hunting (.bak, .old, .tmp)
- [ ] Configuration file searches
- [ ] API endpoint discovery
- [ ] Admin panel location
- [ ] Development/staging environment detection

Content Discovery Template:

```
Directories Found:
- /admin [STATUS] - [DESCRIPTION]
- /api [STATUS] - [DESCRIPTION]
- /backup [STATUS] - [DESCRIPTION]

Files of Interest:
- /robots.txt - [FINDINGS]
- /sitemap.xml - [ENDPOINTS]
- /.env - [ACCESSIBLE Y/N]

API Endpoints:
- /api/v1/users
- /api/v1/auth
```

Phase 3: Parameter Discovery 🔗

Parameter Enumeration

- [ ] URL parameter discovery
- [ ] POST parameter identification
- [ ] Hidden form field analysis
- [ ] Cookie parameter review

- [ ] Header parameter testing
- [ ] JSON/API parameter mapping

Parameter Template:

```
GET Parameters:
- id: [INTEGER] - User/object identifier
- search: [STRING] - Search functionality
- redirect: [URL] - Redirect parameter

POST Parameters:
- username: [STRING]
- password: [STRING]
- csrf_token: [TOKEN]

Interesting Parameters:
- debug: [BOOLEAN] - Debug mode toggle
- admin: [BOOLEAN] - Admin access flag
```

Phase 4: Vulnerability Assessment 🔓

Input Validation Testing

- [ ] Cross-Site Scripting (XSS)
  - [ ] Reflected XSS in parameters
  - [ ] Stored XSS in user inputs
  - [ ] DOM-based XSS
- [ ] SQL Injection
  - [ ] Error-based injection
  - [ ] Boolean-based blind injection
  - [ ] Time-based blind injection
- [ ] Command Injection
- [ ] Path Traversal
- [ ] File Upload vulnerabilities
- [ ] XXE (XML External Entity)

Authentication & Session Management

- [ ] Weak password policies
- [ ] Session fixation
- [ ] Session hijacking possibilities
- [ ] Brute force protection
- [ ] Password reset vulnerabilities
- [ ] Multi-factor authentication bypass

Business Logic Testing

- [ ] Race conditions
- [ ] Price manipulation
- [ ] Privilege escalation
- [ ] Workflow bypass
- [ ] Rate limiting bypass
- [ ] Payment processing flaws

Vulnerability Testing Template:

```
Vulnerability: [TYPE]
Location: [URL/PARAMETER]
Method: [GET/POST/etc.]
Payload: [PAYLOAD_USED]
Response: [INTERESTING_RESPONSE]
Impact: [BUSINESS_IMPACT]
Reproducible: [Y/N]

Test Results:
✅ - Vulnerable
❌ - Not vulnerable
⚠️ - Needs more testing
```

📊 Finding Documentation System

Critical Finding Template 🔴

```
=== CRITICAL VULNERABILITY ===
Title: [CLEAR_TITLE]
Asset: [EXACT_URL]
Type: [VULN_TYPE]
Discovered: [DATE_TIME]

Impact:
- Immediate business risk
- Data exposure potential
- System compromise possible

Reproduction Steps:
1. Navigate to [URL]
2. [EXACT_STEPS]
3. Observe [RESULT]

Evidence:
```

```
- Screenshot: [FILENAME]
- Request/Response: [DETAILS]
- Video: [IF_APPLICABLE]

Business Impact:
[WHY_THIS_MATTERS_TO_BUSINESS]
```

Standard Finding Template 📝

```
Title: [VULNERABILITY_NAME]
Severity: [CRITICAL/HIGH/MEDIUM/LOW]
Asset: [AFFECTED_URL]
Parameter: [VULNERABLE_PARAMETER]

Description:
[TECHNICAL_DESCRIPTION]

Steps to Reproduce:
1.
2.
3.

Expected Result:
[WHAT_SHOULD_HAPPEN]

Actual Result:
[WHAT_ACTUALLY_HAPPENS]

Impact:
[SECURITY_IMPLICATIONS]

Remediation:
[SUGGESTED_FIX]
```

🗓️ Organized Workflow Schedule

Daily Bug Hunting Routine

```
Morning Setup (30 mins):
- [ ] Review active targets
- [ ] Check for program updates
- [ ] Plan today's focus area
- [ ] Set up testing environment

Core Hunting (4-6 hours):
- [ ] 1 hour: Automated recon
```

```
- [ ] 2-3 hours: Manual testing
- [ ] 1 hour: Deep dive on interesting findings
- [ ] 1 hour: Documentation and reporting

Evening Wrap-up (30 mins):
- [ ] Document all findings
- [ ] Update progress checklist
- [ ] Plan tomorrow's priorities
- [ ] Back up important data
```

Weekly Review Process

```
Every Sunday:
- [ ] Review all findings from the week
- [ ] Analyze what worked vs. what didn't
- [ ] Update methodology based on learnings
- [ ] Plan upcoming week's targets
- [ ] Clean up workspace and files
- [ ] Research new techniques/tools
```

🎯 Target Priority Matrix

High Priority Targets 🚀

- Recently launched features
- User input handling areas
- Authentication mechanisms
- File upload functionality
- API endpoints
- Admin panels

Medium Priority Targets ⚡

- Static content areas
- Information disclosure points
- Configuration files
- Error page analysis
- Third-party integrations

Low Priority Areas 📋

- Public marketing pages
- Static documentation
- Already well-tested components

- Out-of-scope adjacent areas

🔄 Failure Analysis Framework

Common Failure Patterns ❌

Recon Failures

- [ ] Missed obvious subdomains
- [ ] Ignored certificate transparency
- [ ] Skipped GitHub reconnaissance
- [ ] Overlooked API documentation
- [ ] Didn't check for mobile apps

Testing Failures

- [ ] Tested same injection repeatedly
- [ ] Ignored encoding/filtering
- [ ] Missed context-specific payloads
- [ ] Didn't consider application logic
- [ ] Forgot about different HTTP methods

Documentation Failures

- [ ] Poor screenshot quality
- [ ] Missing reproduction steps
- [ ] Unclear impact explanation
- [ ] No business context provided
- [ ] Disorganized evidence

Learning Template

```
Date: [DATE]
Target: [TARGET]
What Failed: [SPECIFIC_FAILURE]
Why It Failed: [ROOT_CAUSE]
Lesson Learned: [KEY_INSIGHT]
Prevention Strategy: [HOW_TO_AVOID]
```

📚 Manual Mode Workflow

> *This is "manual mode" - it works, but requires discipline and organization*

1. Setup Your Workspace 📁

Create this folder structure:

```
BugBounty_Workspace/
├── Active_Targets/
├── Completed_Targets/
├── Templates/
├── Tools_and_Scripts/
├── Learning_Notes/
└── Reports_Ready/
```

2. Use Physical/Digital Notebooks 📝

- Keep a master target list
- Maintain daily hunting logs
- Track methodology improvements
- Record interesting techniques found

3. Follow the Checklists ✅

- Print out recon checklist
- Use finding templates consistently
- Follow systematic workflow
- Don't skip documentation steps

4. Time Management ⏰

- Set specific time blocks for each phase
- Use pomodoro technique for focus
- Don't get lost in rabbit holes
- Regular progress reviews

🌟 Success Transformation

From Beginner Chaos 😵

- Random scanning hoping for magic
- Notes scattered everywhere
- Forgetting what you've tested
- Reporting out-of-scope findings
- Getting lost in recon forever

To Professional Hunter 🎯

- Systematic approach with clear priorities
- Organized documentation and evidence
- Understanding of application logic
- Clean, professional reports
- Efficient workflow that scales

💡 Community Integration Tips

Learning Acceleration 🚀

- Join bug bounty Discord servers
- Follow successful hunters on Twitter
- Read disclosed reports religiously
- Participate in CTFs and practice labs
- Share knowledge and ask questions

Networking Strategy 🤝

- Contribute to community discussions
- Share interesting techniques (responsibly)
- Help newcomers when you can
- Build relationships with other hunters
- Stay updated on new methodologies

🏅 Final Success Framework

The Mindset Shift 🧠

1. Think Like the Application - Ask "Where would I screw up?"
2. Embrace Failure - Every failed payload teaches something
3. Prioritize Impact - Focus on what matters to the business
4. Document Everything - Your future self will thank you
5. Stay Systematic - Chaos kills productivity

Key Success Metrics 📊

- Time from recon to first finding
- Report acceptance rate
- Severity distribution of findings
- Learning velocity and skill growth

- Community engagement and reputation

Remember: This free template gives you the structure and systematic approach. The difference between finding nothing and finding critical bugs often comes down to organization, patience, and following a proven methodology.

Start here, master the basics, then level up! 🚀