DVWA

### What is DVWA?

DVWA stands for Damn Vulnerable Web Application, which is an open-source web application intentionally built with security flaws, making it "damn vulnerable." It's written in PHP and uses a MariaDB (or MySQL) database, simulating a real website but with deliberate weaknesses. The core idea is to create a legal, controlled environment where we can experiment with hacking techniques without breaking laws or harming actual systems.

Key terms to know:

- **Vulnerability**: A weakness in software that can be exploited, often due to poor coding practices like not validating user inputs or insecure data storage.
- **Exploitation**: The act of using a vulnerability to gain unauthorized access, steal data, or disrupt services.
- **Pentesting (Penetration Testing)**: Ethical hacking to find and fix vulnerabilities before real attackers do.
- **OWASP Top 10**: A list of the most common web vulnerabilities (e.g., SQL Injection, XSS) that DVWA is based on. OWASP (Open Web Application Security Project) is a nonprofit that promotes secure coding.

DVWA lets you toggle difficulty levels (Low, Medium, High, Impossible) for each vulnerability, starting easy for beginners and getting harder to mimic real-world defenses.

## Working Procedure: How to Use It

To use DVWA, you typically follow this workflow:

1. **Deployment:** Run it locally (XAMPP/WAMP) or via a Docker container.
2. **Configuration:** Set the security level (start with "Low").
3. **Exploitation:** Use tools like **Burp Suite** or a browser to find a flaw (e.g., SQL Injection).
4. **Analysis:** Click the "View Source" button to see the "bad" code.
5. **Comparison:** Change the security level to "Impossible" and view the source again to see the "fix."

**History of DVWA:**

DVWA has evolved as a staple in cybersecurity education, reflecting the growth of web security awareness.

- **Past (Origins and Early Development)**: Created in 2008 by Ryan "ethicalhack3r" Dewhurst, a security researcher from Dewhurst Security. The first public release was around 2009, initially as a simple PHP/MySQL app to demonstrate common flaws. Early versions focused on basic vulnerabilities like SQL injection and XSS, inspired by rising web attacks in the 2000s (e.g., after major breaches like the 2007 TJX hack). It was hosted on platforms like SourceForge before moving to GitHub in 2013. Back then, tools like WebGoat (launched in 2002 by OWASP) were among the first, but DVWA stood out for its simplicity and PHP focus, which matched many real websites.
- **Present (Current Tools and Usage)**: Maintained on GitHub by the DVWA team (led by digininja), with over 795 commits and 61 contributors. The latest version includes updates from 2025, like a Broken Access Control module and Vulnerable APIs (released January 29, 2025), plus Docker enhancements for easy setup. It's widely used with modern tools like Burp Suite for intercepting traffic, SQLMap for automated SQL injection testing, and ZAP (OWASP Zaproxy) for scanning. In 2026, it's integrated into cloud setups (e.g., AWS labs) and supports MariaDB/SQLite3 for broader compatibility. Prebuilt Docker images from GitHub Container Registry make it quick to deploy, reflecting the shift to containerized learning environments.
- **Future Outlook**: As web tech advances, DVWA is likely to incorporate more API-focused vulnerabilities (e.g., RESTful APIs, GraphQL), serverless exploits, and AI-related risks like prompt injection in web-integrated LLMs. Trends suggest integration with VR/AR for immersive pentesting simulations or automated vuln generators using AI. With rising cyber threats (e.g., projected 15 million cybersecurity jobs by 2030), tools like DVWA will remain essential for training, possibly evolving into collaborative platforms for global CTFs (Capture The Flag challenges).

**Uses of DVWA: Why It's Valuable**

DVWA is primarily for education and ethical practice:

- **Learners/Students**: Understand how vulnerabilities arise from code mistakes, like not sanitizing inputs.
- **Developers**: Learn secure coding by seeing (and fixing) flaws in action.
- **Security Pros**: Test tools and techniques legally, e.g., practicing exploits in a sandbox.
- **Teachers**: Classroom demos to show real-time hacks and defenses.
- **Broader Applications**: In certifications like CEH (Certified Ethical Hacker) or OSCP (Offensive Security Certified Professional), where hands-on labs are key. It's also used in bug bounty programs to simulate targets.

Unlike real hacking, DVWA ensures everything is contained— no risk to external systems.

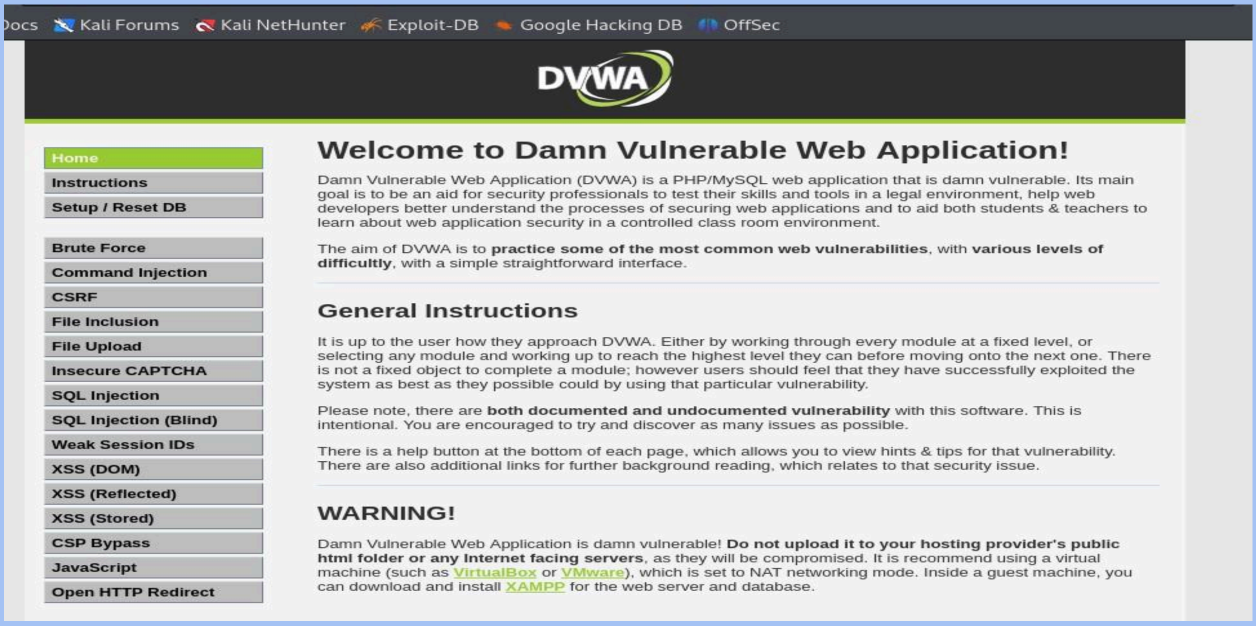
**How DVWA Works: Working Procedure and Setup**

Setting up and using DVWA is straightforward, emphasizing hands-on learning.

1. **Installation (Step-by-Step)**:
   - **Prerequisites**: PHP 7.3+, MariaDB/MySQL, Apache (or XAMPP for Windows).
   - **Via Docker (Recommended for Beginners)**: Clone the GitHub repo, run `docker compose up -d`, and access at `http://localhost:4280`.
   - **Manual Setup**: Download from GitHub, copy to your web server (e.g., `/var/www/html/` on Linux), configure `config.inc.php` with DB credentials (default: user 'dvwa', pass 'p@ssw0rd'), make folders writable (e.g., `hackable/uploads/`), and create the database via the web interface by clicking "Create/Reset Database."
   - **Troubleshooting**: Enable `mod_rewrite` in Apache, set PHP `allow_url_include` to On for certain vulns.
2. **Using It**:
   - Login (default: admin/password).
   - Choose a vulnerability module (e.g., SQL Injection).
   - Set difficulty: Low = no protections; High = some filters.
   - Exploit: Enter malicious inputs to trigger flaws, view source code to see why it fails.
   - Learn: Each module has a "View Source" button showing vulnerable PHP code.

This procedure mirrors real pentesting: reconnaissance, exploitation, analysis.

Here's what the DVWA interface looks like for context:



**Welcome to Damn Vulnerable Web Application!**

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

**General Instructions**

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

**WARNING!**

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

**Differences with Other Tools**

DVWA isn't the only vulnerable app; here's how it compares to popular alternatives:

| Tool | Base Tech | Key Focus | Difficulty Levels | Differences from DVWA | Best For |
|---|---|---|---|---|---|
| **DVWA** | PHP/MariaDB | Common web vulns (e.g., SQLi, XSS, File Upload) with undocumented flaws | Low-Medium-High-Impossible | Simple, PHP-centric; encourages code review and discovery | Quick PHP-based learning, ethical tool testing |
| **WebGoat** | Java/Spring | OWASP Top 10 lessons with guided challenges | Beginner-Advanced | More structured tutorials, Java-focused; includes access control and crypto vulns; less "damn vulnerable," more educational paths | In-depth lessons for developers |
| **OWASP Juice Shop** | Node.js/Express | Modern web (APIs, JWT, NoSQL injection) in a CTF-style e-commerce site | Easy-Hard (star-based) | JavaScript-heavy, realistic scenarios; over 100 challenges; focuses on frontend/backend integration | CTF enthusiasts, modern stack training |
| **bWAPP (bee-box)** | PHP/MySQL | Over 100 vulns, including rare ones like Heartbleed | Low-High | Broader vuln coverage; comes as a VM (bee-box); more enterprise-like | Comprehensive testing, VM-based labs |

DVWA is lighter and faster to set up than VM-heavy tools like Metasploitable (which targets OS/network vulns, not just web).

**Causing Effects: Real-World Impacts of Poor Coding. Vulnerabilities in DVWA stem from bad practices, like not escaping user inputs or weak authentication. In real apps, these lead to devastating effects:**

- **Data Breaches**: SQL injection (as in DVWA) allowed hackers to dump databases in the 2017 Equifax breach, exposing 147 million people's data.
- **Financial Loss**: XSS can steal cookies/sessions, leading to account takeovers; e.g., the 2020 Twitter hack via XSS-like flaws cost millions in scams.
- **Reputation Damage**: Unpatched file uploads (DVWA example) enable malware uploads, as in the 2016 Panama Papers leak.
- **Legal/Compliance Issues**: GDPR fines for breaches can reach €20 million.
- **Broader Risks**: Chain attacks, where one vuln leads to full system compromise, amplifying damage.

Fixing these (e.g., using prepared statements in PHP) prevents hacks, saving billions—cybercrime costs hit $8 trillion globally in 2023, projected to $13.8 trillion by 2028.

**Trends in Deliberately Vulnerable Apps**

- **Containerization Boom**: Docker/Kubernetes setups (as in DVWA's 2025 updates) make labs portable and scalable.
- **API and Modern Vulns**: Shift from traditional web to API security (DVWA's new module), reflecting cloud-native apps.
- **AI Integration**: Tools auto-generating vulns or simulating attacks; DVWA may add ML-related flaws by 2030.
- **Community and Open-Source Growth**: More multilingual support (DVWA has READMEs in 10+ languages) and CTF events; usage spiked 30% in education post-2020 remote learning.
- **Ethical Focus**: Emphasis on "hack responsibly" with legal disclaimers, amid rising regulations.

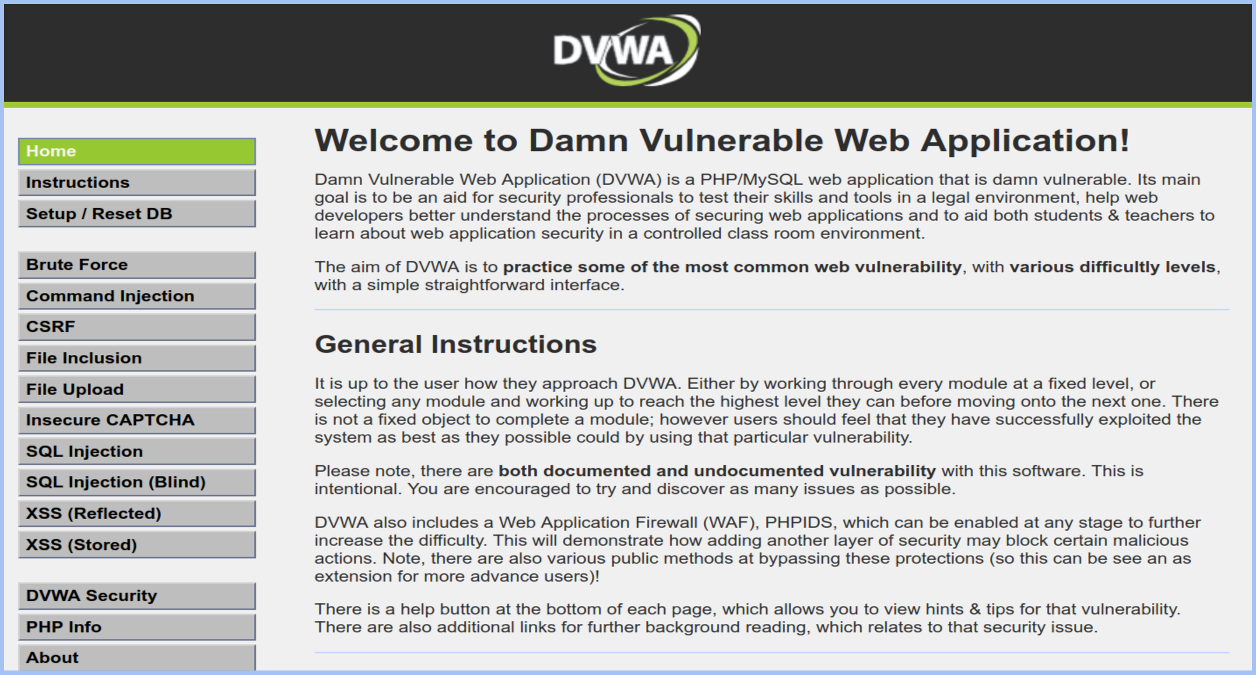
**Examples: Hands-On with DVWA**

Let's walk through two common vulnerabilities to see poor coding in action.

1. **SQL Injection (SQLi)**:
   - **How It Works**: In Low difficulty, DVWA uses unsanitized inputs in queries like `SELECT * FROM users WHERE id = '$id';`. Enter `' OR '1'='1` to bypass checks and dump all users.
   - **Poor Coding Cause**: No prepared statements or escaping.
   - **Exploitation**: Use browser dev tools or SQLMap: `sqlmap -u "http://localhost/vulnerabilities/sqli/?id=1&Submit=Submit" --dump`.
   - **Real Effect**: Could steal passwords; fix with PDO prepared statements.
2. **Cross-Site Scripting (XSS) Reflected**:
   - **How It Works**: Input `<script>alert('Hacked!')</script>` in a search field; it executes on page load due to no output encoding.
   - **Poor Coding Cause**: Echoing user input directly in HTML.
   - **Exploitation**: Craft a malicious URL to steal cookies; in DVWA, view source to see `echo $_GET['name'];`.
   - **Real Effect**: Phishing or session hijacking; fix with `htmlspecialchars()`.

Try these in your setup—start on Low, analyze code, then ramp up difficulty.

For a visual of the SQLi module:



## Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerability**, with **various difficultly levels**, with a simple straightforward interface.

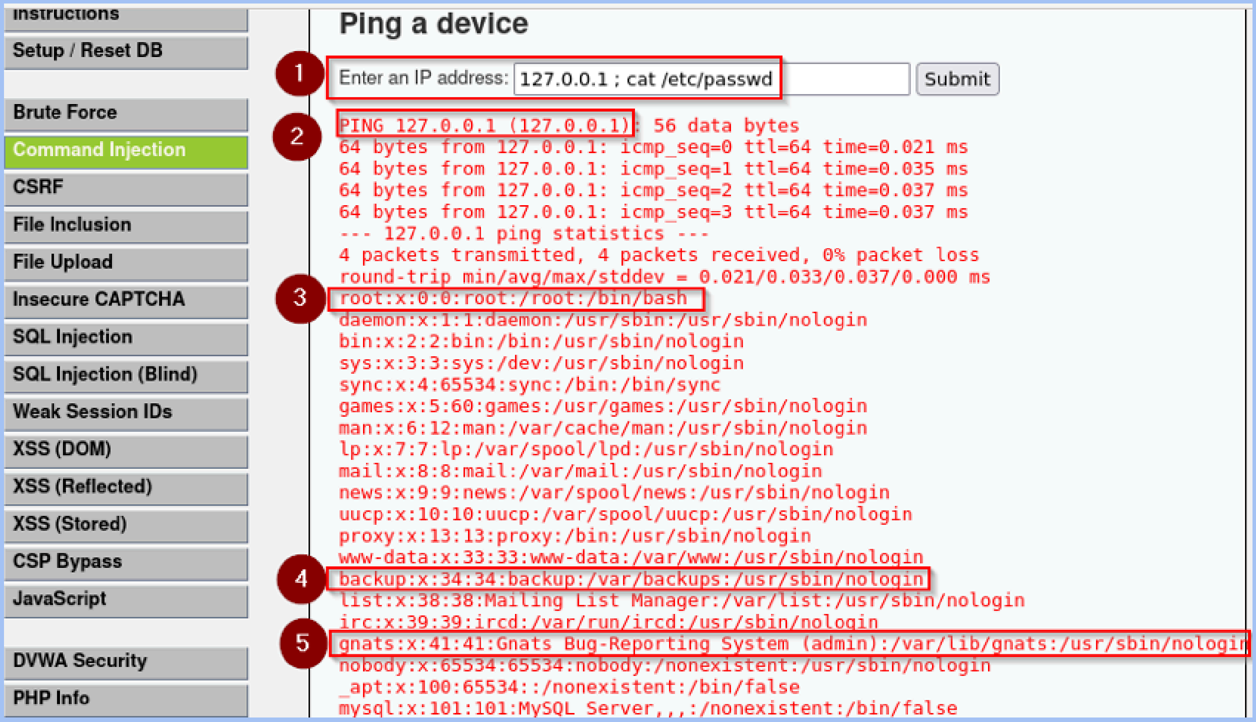
### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be see an as extension for more advance users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.



Ping a device

Enter an IP address: 127.0.0.1 ; cat /etc/passwd

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.021 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.021/0.033/0.037/0.000 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

**Overview:**

DVWA is an essential, free tool for demystifying web vulnerabilities through intentional flaws in PHP code. From its 2008 origins as a simple demo to 2026's Docker-enhanced versions with API modules, it's grown to support ethical learning, tool testing, and secure development. Unlike more structured (WebGoat) or modern (Juice Shop) alternatives, DVWA's strength is its raw, discovery-based approach. Poor coding shown here causes real hacks, but practicing fixes builds better habits.