Execution Risks

Execution Risks

- PHP Execution in Uploads
- Direct PHP Access Allowed

PHP Execution in Uploads

What This Means

This finding indicates that PHP files can be executed within upload or storage directories.

These directories are typically intended for file storage, not code execution.

Why It Matters

Upload directories (such as `/wp-content/uploads/`) are commonly writable by the application.

If PHP execution is allowed in these locations, attackers may:

- upload malicious scripts
- execute code through vulnerable upload points
- gain unauthorized access or control

This is a common method used in real-world WordPress attacks.

How Steel Security Detects This

Steel Security checks whether PHP files placed in upload directories can be executed.

If execution is possible, the directory is flagged as a risk.

How to Fix It

To resolve this issue:

- disable PHP execution in upload and storage directories
- apply server-level rules to prevent script execution
- use Steel Security hardening controls related to execution restrictions

---

What to Expect After Fixing

After applying protections:

- PHP files in upload directories will not execute
- uploaded scripts will be treated as files, not code
- your site will be significantly more resistant to upload-based attacks

---

How to Verify

To verify the fix:

1. upload or place a test PHP file in an upload directory
2. attempt to access it via browser
3. confirm that execution is blocked

Expected results include:

- file download instead of execution
- or a blocked request (e.g., 403 Forbidden)

Common Causes

- default server configuration allowing execution
- missing directory-level restrictions
- insecure file upload handling

Best Practices

- never allow PHP execution in upload directories
- treat storage locations as non-executable
- regularly review upload behavior and permissions
- combine with other execution and file protection controls

Related

- Prevent Execution in Sensitive Directories
- Block Direct PHP Access
- Restrict Sensitive Endpoints

Direct PHP Access Allowed

What This Means

This finding indicates that certain PHP files on your site can be accessed directly via a browser.

These files may not be intended to be executed outside of normal WordPress workflows.

Why It Matters

Many PHP files are designed to be included internally, not accessed directly.

If these files are accessible, attackers may:

- execute scripts in unintended ways
- bypass application logic
- probe for vulnerabilities
- trigger unintended behavior

Restricting direct access reduces these risks.

How Steel Security Detects This

Steel Security identifies PHP files that should not be directly accessible and checks whether they can be executed via a browser.

If direct access is possible, the file is flagged.

How to Fix It

To resolve this issue:

- restrict direct access to sensitive PHP files using server rules
- ensure scripts are only executed through WordPress entry points

- apply Steel Security hardening controls related to PHP access restrictions

---

What to Expect After Fixing

After applying protections:

- direct requests to restricted PHP files will be blocked
- unauthorized execution attempts will fail
- normal site functionality should remain unchanged

How to Verify

To verify the fix:

1. attempt to access a flagged PHP file directly via its URL
2. confirm that access is denied (e.g., 403 Forbidden)

Ensure that:

- restricted files are not accessible directly
- legitimate site functionality continues to work

Common Causes

- plugin or theme files exposed without access restrictions
- custom scripts placed in public directories
- missing or incomplete server rules

Best Practices

- avoid exposing internal PHP files directly
- route execution through WordPress where possible
- review custom scripts and endpoints carefully
- combine with other execution controls

Related

- Block Direct PHP Access
- PHP Execution in Uploads
- Restrict Sensitive Endpoints