When writing Python scripts for automation, several common code vulnerabilities can occur, especially when handling untrusted input, external resources, or system operations. Here’s a list of some of the most relevant vulnerabilities you should be aware of:

1. Path Traversal

- **Description:** Manipulation of file paths to access files/directories outside the intended scope.
- **Example:** Using user input as part of a file path without validation.

2. Shell Injection (Command Injection)

- **Description:** Injecting arbitrary commands into system calls (e.g., os.system, subprocess).
- **Example:** Passing untrusted input to shell commands.

3. Insecure Deserialization

- **Description:** Loading data from untrusted sources using pickle, yaml.load, etc., can lead to code execution.
- **Example:** Using pickle.loads() on user-supplied data.

4. Insecure Use of eval()/exec()

- **Description:** Executing code from untrusted input.
- **Example:** Using eval() or exec() with unsanitized input.

5. Insecure Temporary File Handling

- **Description:** Creating temp files insecurely could allow race conditions or exposure to other users.
- **Example:** Not using the tempfile module securely.

6. Hardcoded Credentials/Secrets

- **Description:** Storing sensitive info in code, which can be leaked.
- **Example:** API keys, passwords hardcoded in scripts.

7. XML External Entity (XXE) Attacks

- **Description:** Parsing XML with vulnerable libraries enables attackers to access local files or network resources.
- **Example:** Using xml.etree.ElementTree without disabling external entity resolution.

8. SQL Injection

- **Description:** Constructing SQL queries directly from user input.
- **Example:** Using string formatting to build SQL statements.

9. Cross-Site Scripting (XSS)

- **Description:** Less common in pure automation scripts, but if output is web-accessible, XSS is possible.

10. Insecure Permissions (Privilege Escalation)

- **Description:** Scripts running with more privileges than required.
- **Example:** Running as root/admin unnecessarily.

11. Race Conditions

- **Description:** Improper handling of file/OS operations in multi-threaded or concurrent scripts.

12. Unvalidated Input

- **Description:** Accepting and using user input without validation or sanitization.
- **Example:** File uploads, network input.

13. Resource Injection

- **Description:** Allowing untrusted input to specify resources (e.g., file names, class names).

14. Information Disclosure

- **Description:** Logging sensitive data or exposing error messages with sensitive info.

15. Use of Outdated Libraries

- **Description:** Using dependencies with known vulnerabilities.

16. Server-Side Request Forgery (SSRF)

- **Description:** If scripts fetch URLs based on user input, attackers may access internal services.

17. Unsafe YAML Loading

- **Description:** Using yaml.load instead of yaml.safe_load.

18. Unrestricted File Upload

- **Description:** Allowing any file type upload, leading to malicious files being executed.

19. Unsafe Pickle Usage

- **Description:** Using pickle to load untrusted data, leading to code execution.

Summary Table

Vulnerability	Description
Path Traversal	Accessing unintended files/directories
Shell Injection	Executing unintended shell commands
Insecure Deserialization	Arbitrary code execution via untrusted serialization
Insecure eval/exec	Code execution from untrusted input

Vulnerability	**Description**
Insecure Temp Files	Race conditions, file exposure
Hardcoded Credentials	Secrets exposed in code
XXE Attacks	Unintended file/network access via XML parsing
SQL Injection	Arbitrary SQL execution
XSS	Web output injection
Insecure Permissions	Running with excessive privileges
Race Conditions	Concurrent execution issues
Unvalidated Input	Unsanitized user input
Resource Injection	Untrusted input as resource identifiers
Info Disclosure	Sensitive info in logs/errors
Outdated Libraries	Vulnerable dependencies
SSRF	Internal network access via user-controlled URLs
Unsafe YAML Loading	Code execution via yaml.load
Unrestricted File Upload	Malicious file execution
Unsafe Pickle Usage	Arbitrary code execution from pickle

Let me know if you want examples or mitigations for any specific vulnerability!