20TH MARCH 2025

ASSIGNMENT

INT 310- CODE VULNERABILITY ANALYSIS/ AMINU IDRIS

SUBMITTED BY:

Ayilara Busari Dare

IDEAS/24/28133

INT310: Web Application Source Code Vulnerability Analysis – Lab 2: Comprehensive Vulnerability Review

Introduction to Code Vulnerability Analysis

Code vulnerability analysis is the process of reviewing source code to identify security flaws that attackers could exploit. It helps prevent cyber threats, ensures compliance with security standards, and strengthens software security.

Common Vulnerabilities in PHP & Python

- Injection Attacks (SQLi, XSS, Command Injection)
- Insecure File Handling (Unrestricted Uploads, Directory Traversal)
- Weak Authentication & Authorization
- Deserialization Issues (Remote Code Execution)

Analysis Methods

1. Manual Code Review – Inspecting code for flaws.
2. Static Analysis Tools – Automated scanning (e.g., SonarQube, Bandit).
3. Dynamic Analysis – Testing the application in a running state.
4. Fuzz Testing – Sending unexpected input to uncover vulnerabilities.

Best Practices for Secure Coding

- Sanitize and validate user input.
- Use parameterized queries to prevent SQLi.
- Implement strong authentication and access controls.
- Restrict file uploads and validate file types.
- Keep dependencies updated.

1. **Review Code**: Analyze the provided PHP and Python code files for security vulnerabilities.
    - **PHP File**: vulnerable_script.php

```
kali@kali: ~/php_analysis
File  Actions  Edit  View  Help
  GNU nano 8.1                                   vulnerable.php *
<?php
// Database connection
$conn = mysqli_connect("localhost", "root", "", "test_db");


$username = $_GET['username'];
$query = "SELECT * FROM users WHERE username = '$username'";
$result = mysqli_query($conn, $query);


$cmd = $_GET['cmd'];
system($cmd);


$admin_user = "admin";
$admin_password = "password123";


echo "Hello, " . $_POST['name'];


if (isset($_GET['delete_user'])) {
    $user_id = $_GET['delete_user'];
    $query = "DELETE FROM users WHERE id = $user_id";
    mysqli_query($conn, $query);

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo
```

```
kali@kali: ~/php_analysis
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/php_analysis]
└─$ nano vulnerable.php


┌──(kali㉿kali)-[~/php_analysis]
└─$ ls
vulnerable.php

┌──(kali㉿kali)-[~/php_analysis]
└─$ grep -E "system|exec|shell_exec|passthru|eval|base64_decode|assert|include|require|file_put_contents|
fopen|fwrite|unlink|chmod" vulnerable.php

system($cmd);
file_put_contents('user_data.txt', $admin_password);
include($path . '/file.php');
chmod("sensitive_data.txt", 0777);

┌──(kali㉿kali)-[~/php_analysis]
└─$
```

unsanitized SQL queries: This is **vulnerable to SQL Injection** because $username is taken directly from $_GET without sanitization

```
┌──(kali㉿kali)-[~/php_analysis]
└─$ grep -i "select\|insert\|update\|delete" vulnerable.php

$query = "SELECT * FROM users WHERE username = '$username'";
if (isset($_GET['delete_user'])) {
    $user_id = $_GET['delete_user'];
    $query = "DELETE FROM users WHERE id = $user_id";
    $query = "UPDATE users SET password = '$new_password' WHERE id = 1";
```

remote command execution: This allows **arbitrary command execution** via URL (?cmd=whoami).

```
┌──(kali㉿kali)-[~/php_analysis]
└─$ grep -E "system|exec|shell_exec|passthru" vulnerable.php

system($cmd);
```

File Inclusion Vulnerabilities:
This is Local File Inclusion (LFI), allowing an attacker to read system files like /etc/passwd.

```
┌──(kali㉿kali)-[~/php_analysis]
└─$ grep -E "include|require" vulnerable.php

include($path . '/file.php');
```

Hardcoded Credentials: Hardcoded passwords are a major security risk.

```
┌──(kali㉿kali)-[~/php_analysis]
└─$ grep -E "password|admin|root" vulnerable.php

$conn = mysqli_connect("localhost", "root", "", "test_db");
$admin_user = "admin";
$admin_password = "password123";
file_put_contents('user_data.txt', $admin_password);
$is_admin = $_GET['is_admin'];
if ($is_admin == "true") {
if (isset($_POST['change_password'])) {
    $new_password = $_POST['new_password'];
    $query = "UPDATE users SET password = '$new_password' WHERE id = 1";
$hash = md5($admin_password);
    echo "Trying password ...";
$hash = sha1($admin_password);
```

Open Redirects: This allows an attacker to redirect users to phishing sites.

```
┌──(kali㉿kali)-[~/php_analysis]
└─$ grep -E "header\(" vulnerable.php

header("Location: " . $_GET['redirect_to']);
```

- **Python File**: vulnerable_script.py

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
  GNU nano 8.1                    vulnerable.py *
import os

user_input = input("Enter a command: ")
os.system(user_input)

file = open('sensitive_data.txt', 'w')
file.write('Secret Data')
file.close()

def calculate(data):
    result = data * 99999999999999999999999999
    print(result)

data = int(input("Enter a number: "))
calculate(data)

buffer = bytearray(10)
data = input("Enter more than 10 characters: ")
buffer[:len(data)] = data.encode()

def admin_action():
File Name to Write: vulnerable.py
^G Help        M-D DOS Format     M-A Append         M-B Backup File
^C Cancel      M-M Mac Format     M-P Prepend        ^T Browse
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nano vulnerable.py

┌──(kali㉿kali)-[~/Desktop]
└─$ cat vulnerable.py

import os

user_input = input("Enter a command: ")
os.system(user_input)

file = open('sensitive_data.txt', 'w')
file.write('Secret Data')
file.close()


def calculate(data):
    result = data * 99999999999999999999999999
    print(result)

data = int(input("Enter a number: "))
calculate(data)
```

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
>> Issue: [B307:blacklist] Use of possibly insecure function - consider using safer ast.literal_eval.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_calls.html#b307-eval
   Location: ./vulnerable.py:50:0
49
50      eval(input("Enter Python code to execute: "))

--------------------------------------------------

Code scanned:
        Total lines of code: 29
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 1
                Medium: 3
                High: 2
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 1
                High: 5
Files skipped (0):
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ bandit vulnerable.py

[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.9
Run started:2025-03-20 11:57:07.210399

Test results:
>> Issue: [B605:start_process_with_a_shell] Starting a process with a shell, possible injection detected,
 security issue.
   Severity: High   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b605_start_process_with_a_shell.html
   Location: ./vulnerable.py:5:0
4       user_input = input("Enter a command: ")
5       os.system(user_input)
6

--------------------------------------------------
>> Issue: [B105:hardcoded_password_string] Possible hardcoded password: 'secret'
   Severity: Low   Confidence: Medium
   CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b105_hardcoded_password_string.html
   Location: ./vulnerable.py:34:17
```

```
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b105_hardcoded_password_string.html
   Location: ./vulnerable.py:34:17
33      admin_username = "admin"
34      admin_password = "secret"
35
--------------------------------------------------
>> Issue: [B102:exec_used] Use of exec detected.
   Severity: Medium   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b102_exec_used.html
   Location: ./vulnerable.py:38:0
37      external_func = input("Enter the function to call: ")
38      exec(external_func)
39
--------------------------------------------------
>> Issue: [B605:start_process_with_a_shell] Starting a process with a shell, possible injection detected,
 security issue.
   Severity: High   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b605_start_process_with_a_shell.html
   Location: ./vulnerable.py:46:0
45      url = "http://example.com/file.py"
46      os.system(f"wget {url} -O downloaded_file.py")
47      exec(open('downloaded_file.py').read())

--------------------------------------------------
>> Issue: [B102:exec_used] Use of exec detected.
```

```
File  Actions  Edit  View  Help
   More Info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_calls.html#b307-eval
   Location: ./vulnerable.py:50:0
49
50      eval(input("Enter Python code to execute: "))

--------------------------------------------------
Code scanned:
        Total lines of code: 29
        Total lines skipped (#nosec): 0
Run metrics:
        Total issues (by severity):
          Undefined: 0
          Low: 1
          Medium: 3
          High: 2
        Total issues (by confidence):
          Undefined: 0
          Low: 0
          Medium: 1
          High: 5
Files skipped (0):
```

(1) COMMAND INJECTION (os.system(user_input))

-The program **directly executes user input as a system command** using os.system(user_input).

-An attacker could enter a malicious command like rm -rf / (deletes all files) or cat /etc/passwd (reads user credentials).

CWE: CWE-78: Improper Neutralization of Special Elements used in an OS Command

Severity: High

Fix: Use subprocess.run() with proper input validation.

(2)**. Hardcoded Password (admin_password = "secret") Issue:**

- The script **stores the admin password in plain text**, making it easy for attackers to steal if they gain access to the source code.

CWE: CWE-259: Use of Hardcoded Password
Severity: Low
Fix: Store passwords securely using environment variables or a secret manager.

(3). Arbitrary Code Execution (exec(external_func)) Issue:

- The exec() function **executes any user input as Python code**, allowing **remote code execution (RCE)**.
- An attacker could input os.system('rm -rf /') to delete system files.

CWE: CWE-78: Improper Neutralization of Special Elements used in an OS Command
Severity: Medium
Fix: Use **whitelisting** and limit allowed function calls.

(4). Insecure File Download (os.system(f"wget {url} -O downloaded_file.py")) Issue:

- The script **downloads a file from an external URL and executes it without verification**.
- If an attacker **modifies the file**, it could run malware.

CWE: CWE-78: Improper Neutralization of Special Elements used in an OS Command
Severity: High
Fix: Validate URLs and check file integrity before execution.

(5). Another Arbitrary Code Execution (exec(open('downloaded_file.py').read())) Issue:

- The script **blindly executes the downloaded Python file**, increasing the risk of running malicious code.

CWE: CWE-78: Improper Neutralization of Special Elements used in an OS Command
Severity: Medium
Fix: Verify file integrity before execution.

(6). Dangerous eval(input()) Execution
Issue:

- The program **executes user input as Python code**, making it vulnerable to **code injection attacks**.
- An attacker could enter __import__("os").system("rm -rf /") to delete files.

CWE: CWE-78: Improper Neutralization of Special Elements used in an OS Command
Severity: Medium
Fix: Use **ast.literal_eval()** to safely evaluate input.

2. **Identify Security Vulnerabilities**:
   - **Document Findings**:

Vulnerability Analysis Worksheet

CWE Number	File Name	Line Number	Description of the Vulnerability
89	vulnerable_script.php	4	SQL Injection vulnerability due to unparameterized query.
78	vulnerable_script.php	9	OS Command Injection via unsanitized user input.
120	vulnerable_script.py	17	Buffer Overflow due to improper handling of user input.
79	vulnerable_script.php	21	Cross-site scripting (XSS) risk from improper input validation.
306	vulnerable_script.php	30	Missing authentication for critical function.
862	vulnerable_script.php	35	Missing authorization check allowing privilege escalation.

798	vulnerable_script.py	34	Hard-coded credentials exposing sensitive information.
311	vulnerable_script.php	40	Missing encryption of sensitive

CWE Number	**File Name**	**Line Number**	**Description of the Vulnerability**
434	vulnerable_script.php	50	data stored in plaintext. Unrestricted file upload leading to potential RCE.
807	vulnerable_script.py	38	Reliance on untrusted user input for execution logic.
250	vulnerable_script.py	46	Execution with unnecessary privileges increasing attack surface.
352	vulnerable_script.php	55	CSRF vulnerability allowing unauthorized actions on behalf of users.
22	vulnerable_script.php	60	Path Traversal vulnerability enabling access to restricted files.
494	vulnerable_script.py	48	Downloading and executing code without integrity check.
863	vulnerable_script.php	65	Incorrect authorization, exposing admin-only functions to users.
829	vulnerable_script.py	50	Inclusion of functionality from an untrusted source.
732	vulnerable_script.php	70	Incorrect permission assignment for critical resources.
			Use of potentially dangerous

676	vulnerable_script.py	52	functions like eval() and exec().
327	vulnerable_script.php	75	Use of a weak cryptographic algorithm (MD5/SHA1).
131	vulnerable_script.py	19	Incorrect calculation of buffer size leading to overflow.
307	vulnerable_script.php	80	Lack of restriction on authentication attempts allowing brute force.

CWE Number	**File Name**	**Line Number**	**Description of the Vulnerability**
601	vulnerable_script.php	85	URL Redirection to an untrusted site through user input.
134	vulnerable_script.py	88	Uncontrolled format string vulnerability.
190	vulnerable_script.py	23	Integer overflow leading to unpredictable results.
759	vulnerable_script.php	90	Use of a one-way hash without salt, making it susceptible to attacks.