# Vulnerability Report

## Vulnerability: File Upload

Vulnerability Information

Vulnerability Name: File Upload  in DVWA Location:

[http://localhost/dvwa/vulnerabilities/fileupload/](http://localhost/dvwa/vulnerabilities/fileupload/)

Severity: High

Vulnerability Description

Summary:

Steps to Reproduce:

1. Login to DVWA: Use the default credentials (admin: admin) to log in
2. **Upload a File**: Click on the "File Upload" option from the vulnerability menu
3. Execute the Uploaded File: After uploading, the file is stored in the hackable/uploads/ directory. Access the file through the provided URL to execute it.

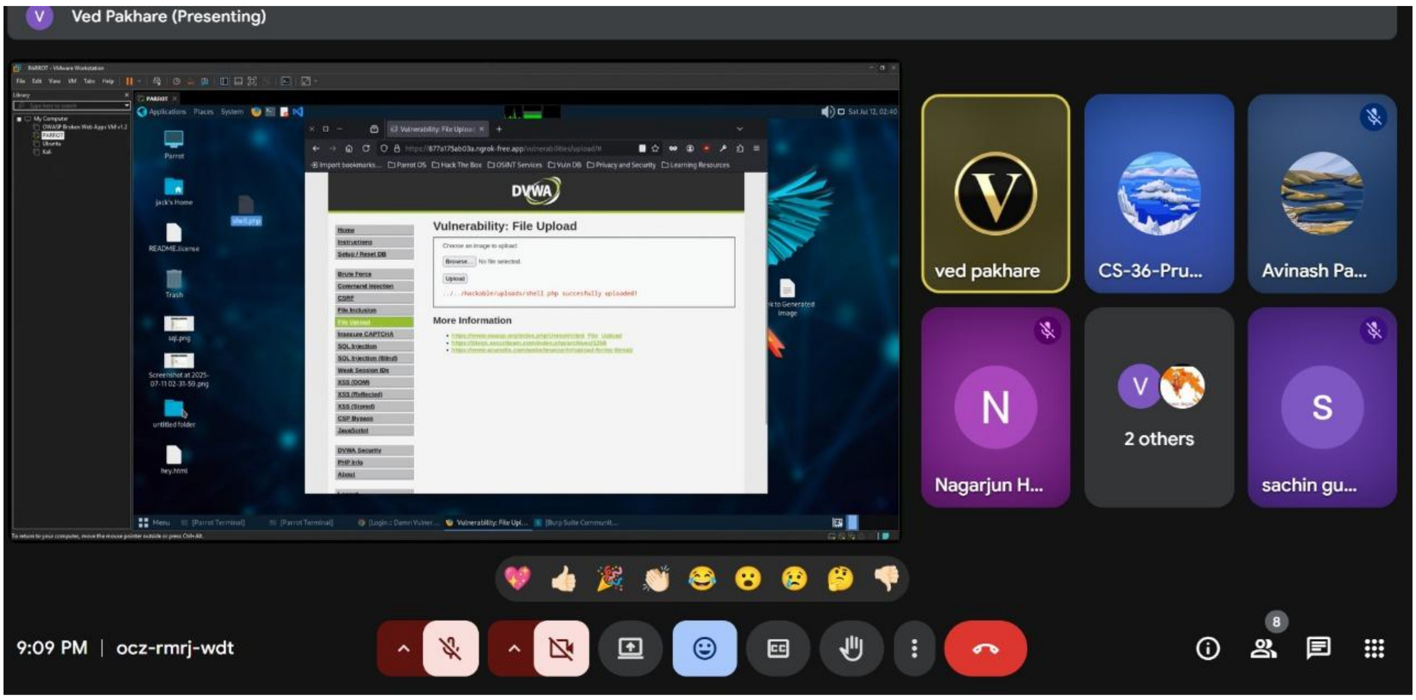
Proof of Concept (PoC)

Description:

To exploit the file upload vulnerability in DVWA, log in (default: admin/password), set security to

low, and go to the File Upload section. Create a malicious PHP file (e.g., using msfvenom),

upload it, and access the uploaded file's URL to execute it, potentially gaining a reverse shell



```
../../hackable/uploads/shell.php succesfully uploaded!
```

Recommendations

Mitigation:

- Validate File Types:
- Secure Storage
- Limit and Scan

Best Practices:

- Apply Allowlist File Extensions.
- Implement web application firewalls.
- Conduct regular security audits.
- Train developers in secure coding.

## Vulnerability: Command Injection

Vulnerability Information

Vulnerability Name: command injection Attack in DVWA

Location: http://localhost/dvwa/vulnerabilities/commandinj/

Severity: Medium

# Vulnerability Report

## Vulnerability Description

### Summary:

Command injection in DVWA stems from poor input validation, enabling attackers to run arbitrary system commands (e.g., ; ls or && whoami). This can lead to unauthorized access or control. Prevent it by sanitizing and validating all user inputs to block malicious command execution.

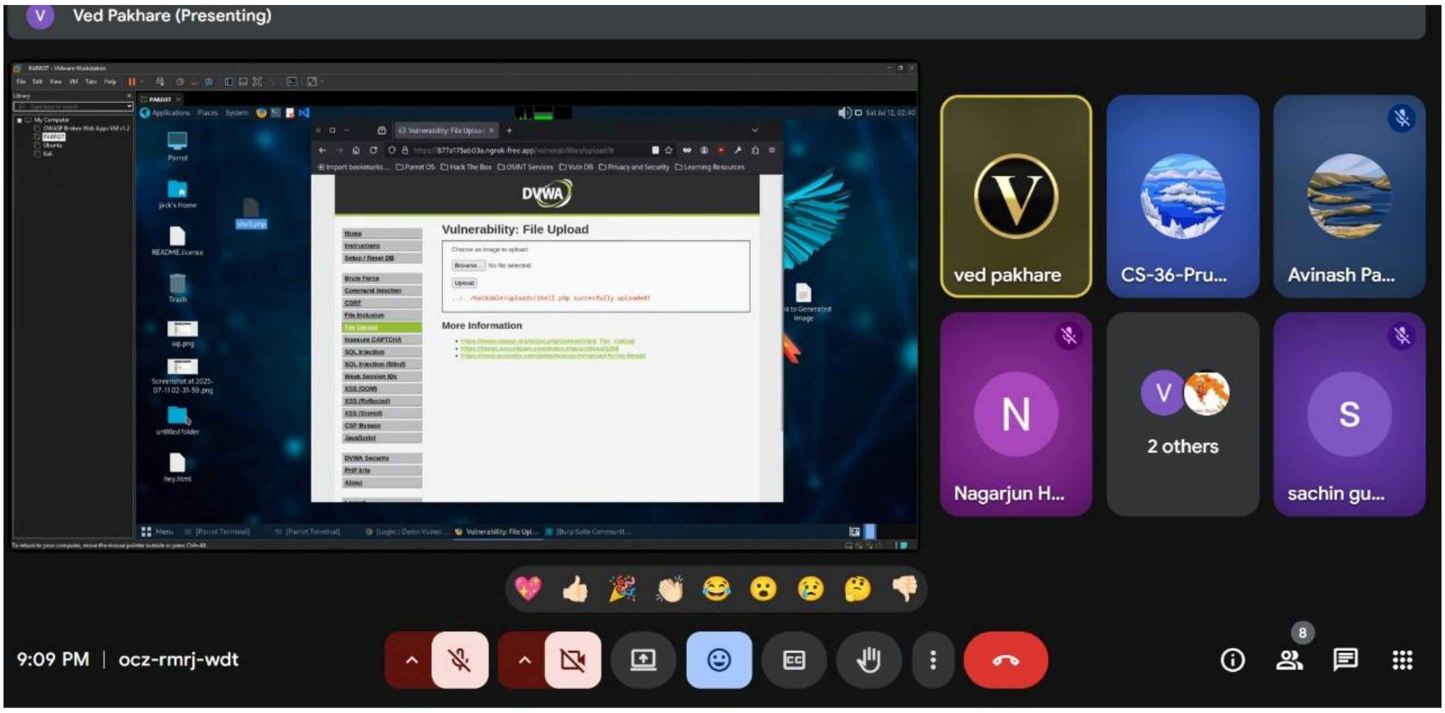
### Steps to Reproduce:

1. navigate to: http://localhost/dvwa/vulnerabilities/commandinj/
2. Inject Basic Command = enter 127.0.0.1; whoami or 127.0.0.1 && dir, then submit.
3. Verify Output : Check the response for system information

## Proof of Concept (PoC)

### Description:

In DVWA, command injection occurs due to unsanitized input fields, allowing attackers to append malicious commands like ; whoami or && dir to execute arbitrary system commands, potentially exposing sensitive data or gaining unauthorized server access.



```
../../hackable/uploads/shell.php succesfully uploaded!
```

Recommendations

Mitigation:

- Validate Inputs: Allow only expected formats (e.g., IP addresses) using regex/whitelisting..
- Sanitize Commands: Use escapeshellcmd() or escapeshellarg() to prevent command execution.
- Restrict Permissions: Run web server with minimal privileges to limit damage.

Best Practices:

- Implement Access Control:
- Use Indirect Object References:.
- Use Security Headers and Framework Protections:

## Team Member Information

Reported by: Sachin Gupta

Reviewed by: Ved Pakhare