Saveetha School of Engineering

Saveetha Institute of Medical and Technical Sciences

Department of Computer Science Engineering

CSA3107 – Cloud Storage and Security for Beginners

Faculty Name: Dr. V. Gokula Krishnan

<u>List of Lab Experiments</u>

1. Perform basic cloud storage operations using Google Drive, including copying files, backing up data, synchronizing across devices, and sharing files.

2. Explore different interfaces for accessing and interacting with Google Drive, including proprietary software clients, browser interfaces, and APIs.

3. Demonstrate cloud storage optimization techniques (Deduplication, Delta Encoding, Compression) using Dropbox, a widely used cloud storage service.

4. Demonstrate cloud security aspects such as registration and login, transport security, encryption, file sharing, multi-device access, update functionality, and server location using MEGA Cloud Storage.

5. Analyze security threats, time-related aspects, advanced persistent threats (APT), and client-server encryption in a cloud storage environment using pCloud, a secure cloud storage provider with built-in encryption features.

6. Analyze file storage, synchronization, sharing, security threats, and encryption mechanisms using CloudMe.

7. Demonstrate file synchronization, backup, multi-device access, and secure file sharing using Dropbox.

8. Demonstrate file synchronization, backup, multi-device access, and encryption using TeamDrive.

9. Demonstrate secure file sharing, collaboration, and permission control using TeamDrive.

10. Simulate a Man-in-the-Middle (MITM) attack on cloud storage traffic and detect it using Wireshark.

11. Test and analyze file versioning in Dropbox, which allows users to recover previous versions of a file after modifications or deletions.

12. Evaluate the upload speed, download speed, response time, and efficiency of different cloud storage services (e.g., Dropbox, Google Drive, OneDrive) using Apache JMeter.

13. Demonstrate how to recover permanently deleted or lost OneDrive cloud files using Recuva tool.

14. Encrypt files before uploading them to Dropbox using VeraCrypt to ensure data security and confidentiality.

15. Encrypt files before uploading them to a cloud storage service using Cryptomator and verify the encryption effectiveness.

16. Automate cloud backup, synchronization, and file transfer between different cloud storage services using Rclone.

17. Monitor and analyze cloud storage traffic using Wireshark on Windows, identifying security risks and traffic patterns.

18. Verify data integrity before and after cloud storage upload/download using HashCalc checksum tool.

19. Configure and test Role-Based Access Control (RBAC) and permission settings in Google Drive to ensure secure file access and management.

20. Configure Duplicati for automatic encrypted cloud backups and test secure data restoration.

21. Demonstrate how to upload, organize, share, and manage files securely using TeraBox Cloud on Windows.

22. Demonstrate how to upload, synchronize, and manage files securely using OpenDrive Cloud on Windows.

23. Demonstrate how to enable and test Two-Factor Authentication (2FA) in TeamDrive for enhanced cloud storage security.

24. Demonstrate how to securely delete files from Dropbox cloud storage using BleachBit, ensuring data cannot be recovered.

25. Demonstrate how to upload, download, and securely share files using Icedrive, a built-in encryption and modern interface cloud storage service.

1. Perform basic Cloud Storage Operations using Google Drive

Aim: Perform basic cloud storage operations using **Google Drive**, including **copying files, backing up data, synchronizing across devices, and sharing files.**

Software & Tools Required:

- **Google Drive account** (free or paid)
- **Google Drive Desktop App** (for sync)
- **Web browser (Chrome/Edge/Firefox)**

Experiment Steps:

Step 1: Copy Files to Google Drive

1. Open Google Drive in a web browser.
2. Click **"New"** → **"File Upload"** and select a file.
3. After upload, **right-click the file** → **"Make a copy"** to duplicate it.
4. Verify that both the original and copied files are available in Google Drive.

Step 2: Backup Files to Google Drive

1. Install **Google Drive for Desktop** on your laptop (Download here).
2. Open the Google Drive application and **sign in** to your Google account.
3. Click **Preferences** → **Add Folder** and select a local folder to back up.
4. Choose **"Back up to Google Drive"** and start the backup process.
5. Verify that the backed-up files appear in Google Drive.

Step 3: Synchronization Across Devices

1. **Modify a file** in the Google Drive folder on your laptop.
2. Check Google Drive on another device (mobile/laptop) to see if the updated file appears automatically.
3. Try creating a file from another device and confirm its presence on your laptop.

Step 4: Sharing Files with Others

1. **Right-click a file** in Google Drive and select **"Share"**.
2. Enter an email ID to share it with a specific person **(View/Edit access)** or generate a public link.

3. Test the sharing feature by opening the link in a different account/browser.
4. Verify if the recipient can access and modify the file (if permitted).

Expected Outcome:

- Files are successfully **copied, backed up, and synchronized** across devices.
- Shared files can be accessed based on the defined permissions.

---

2. Exploring Different Interfaces for Accessing and Interacting with Google Drive

Objective:

To explore different **interfaces** for accessing and interacting with **Google Drive**, including **proprietary software clients, browser interfaces, and APIs**.

Software & Tools Required:

- **Google Drive Account**
- **Google Drive for Desktop (Proprietary Software Client)**
- **Web Browser (Chrome/Firefox/Edge)**
- **Google Drive API (Using Python & Google Drive API)**

Experiment Steps:

Step 1: Accessing Google Drive via Browser Interface

1. Open **Google Drive** in a web browser (drive.google.com).
2. Click **"New" → "File Upload"** to upload a test file.
3. Verify that the file is accessible and editable via the browser.
4. Try **renaming, moving, and deleting** the file using the browser interface.

Step 2: Using Proprietary Software Client (Google Drive for Desktop)

1. Install **Google Drive for Desktop** (Download here).
2. Open the app and **sign in** with your Google account.
3. Enable **file synchronization** by selecting a folder to sync.
4. Copy a file into the synced folder on your **laptop** and verify that it appears in Google Drive (browser).
5. Modify the file from another device and check if it updates automatically on the laptop.

Step 3: Accessing Google Drive via API (Using Python)

1. Install **Google Drive API library** in Python:

   ```
   bash
   CopyEdit
   pip install google-auth google-auth-oauthlib google-auth-httplib2
   google-api-python-client
   ```

2. Authenticate with Google Drive API (follow Google API setup instructions).

3. Run a Python script to **list files** in Google Drive:

```
python
CopyEdit
from googleapiclient.discovery import build
from google.oauth2 import service_account

# Load credentials
SERVICE_ACCOUNT_FILE = 'path/to/your/service-account.json'
SCOPES = ['https://www.googleapis.com/auth/drive']

creds =
service_account.Credentials.from_service_account_file(SERVICE_ACCOUNT_F
ILE, scopes=SCOPES)
drive_service = build('drive', 'v3', credentials=creds)

# List files
results = drive_service.files().list().execute()
files = results.get('files', [])
for file in files:
    print(f"File: {file['name']}, ID: {file['id']}")
```

4. Run the script and verify that it lists the files in your Google Drive.

Expected Outcome:

- **Browser Interface:** Upload, manage, and edit files via the web.
- **Proprietary Software Client:** Automatic synchronization between local and cloud storage.
- **API:** Programmatic access to Google Drive for automation and file management.

---

3. Cloud Storage Optimization Techniques (Deduplication, Delta Encoding, Compression) Using Dropbox

Objective:

To demonstrate **cloud storage optimization techniques** (Deduplication, Delta Encoding, Compression) using **Dropbox**, a widely used cloud storage service.

Software & Tools Required:

- **Dropbox Account** (Sign up here)
- **Dropbox Desktop App** (For Sync & Delta Encoding)
- **Duplicati** (For Backup & Deduplication) → Download Here
- **7-Zip / WinRAR** (For Compression)

Experiment Steps:

Step 1: Deduplication in Dropbox

1. **Upload a file** (e.g., `report.pdf`) to Dropbox via the web or desktop app.
2. **Duplicate the file** on your local machine (`copy_report.pdf`).
3. **Upload the duplicate file** to Dropbox and observe that the upload completes **instantly** (Dropbox does not store duplicate copies but creates a reference to the original).
4. **Test with Duplicati:**
    - Install **Duplicati** and configure **Dropbox** as a backup destination.
    - Backup the same file multiple times and verify that **only one copy is stored** due to deduplication.

Step 2: Delta Encoding with Dropbox

1. **Upload a large text file** (`data.txt`) to Dropbox.
2. **Modify only a small part** of `data.txt` and save the changes.
3. **Dropbox will not re-upload the entire file** but only the changed portion using **delta encoding**.
4. Check Dropbox **sync speed**—it should be significantly faster compared to a full re-upload.

Step 3: File Compression Before Uploading

1. Select multiple large files (`file1.txt`, `file2.txt`, `file3.txt`).
2. **Compress them** into a ZIP archive using 7-Zip or WinRAR.
3. Upload both **original** and **compressed** versions to Dropbox.
4. Compare:
    - **Storage space used** (compressed vs. uncompressed).

- **Upload time** for compressed vs. individual files.

Expected Outcome:

- **Deduplication:** Dropbox prevents redundant file storage.
- **Delta Encoding:** Only modified parts of a file are updated, saving bandwidth.
- **Compression:** Reduces storage space and speeds up file transfers.

---

4. Cloud Security from a Security Engineer's Perspective using MEGA Cloud Storage

Objective:

To demonstrate **cloud security aspects** such as **registration and login, transport security, encryption, file sharing, multi-device access, update functionality, and server location** using **MEGA Cloud Storage**, which offers built-in end-to-end encryption.

Cloud Service Used:

□ **MEGA Cloud Storage** (Sign up here)

- Offers **zero-knowledge encryption** (end-to-end encryption by default).
- Supports **secure file sharing** with encryption keys.
- Available on **multiple devices** (PC, mobile, browser).

Software & Tools Required:

- **MEGA Cloud Account**
- **MEGA Desktop & Mobile Apps** (for multi-device access)
- **Wireshark (optional, for transport security analysis)**

Experiment Steps:

Step 1: Registration and Login Security

1. Go to MEGA and **create an account** with an email and a strong password.
2. Verify that **two-factor authentication (2FA)** can be enabled for additional security.
3. Try logging in from another browser/device to observe authentication behavior.

Step 2: Transport Security Analysis (HTTPS Encryption)

1. Open **MEGA Web App** and log in.
2. Check the browser address bar—verify that MEGA uses **HTTPS (SSL/TLS encryption)** for secure communication.
3. (Optional) Use **Wireshark** to capture packets and confirm that data is encrypted in transit.

Step 3: File Encryption & Secure Sharing

1. **Upload a file** (`confidential.pdf`) to MEGA.
2. Right-click the file and select **“Get Link”** → Observe that the link includes an **encryption key**.
3. Share the file without the key and verify that the recipient **cannot access it** without the decryption key.

Step 4: Multi-Device Access

1. Install **MEGA Desktop Client** on a PC/laptop and sync files.
2. Install **MEGA Mobile App** and log in.
3. Upload a file from one device and verify that it syncs across all devices automatically.

Step 5: Update Functionality & Server Location

1. Check **MEGA’s server location** under Account → Settings.
2. Observe how MEGA **automatically updates files** across devices when modified.

Expected Outcome:

□ **Secure Registration/Login:** Supports password protection & 2FA.
□ **Transport Security:** Uses SSL/TLS (HTTPS) for secure communication.
□ **Encryption:** Provides built-in end-to-end encryption for file security.
□ **File Sharing:** Secure links with encryption keys prevent unauthorized access.
□ **Multiple Devices:** Ensures file synchronization across PC, browser, and mobile.
□ **Update Functionality:** Real-time updates across devices.
□ **Server Location:** Data stored in privacy-focused jurisdictions (e.g., New Zealand).

---

5. Security Analysis of Cloud Storage Devices using pCloud

Objective:

To analyze **security threats, time-related aspects, advanced persistent threats (APT), and client-server encryption** in a cloud storage environment using **pCloud**, a secure cloud storage provider with built-in encryption features.

Cloud Service Used:

☐ **pCloud** (Sign up for Free)

- Provides **client-side encryption** (with pCloud Crypto).
- Stores files in secure **EU or US-based servers**.
- Supports **zero-knowledge encryption** (for premium users).

Software & Tools Required:

- **pCloud Web Interface**
- **pCloud Desktop App (for Client-Server Encryption Analysis)**
- **Wireshark (for Network Traffic Monitoring, optional)**

Experiment Steps:

Step 1: Analysis of Cloud Storage Threats

1. **Upload a file** (`confidential.txt`) to **pCloud**.
2. **Share the file** using a public link and verify that **anyone with the link can access it** (potential data leakage).
3. **Enable password protection** for the shared link and verify that unauthorized users **cannot access it**.

Step 2: Time-Related Aspects of Cloud Security

1. **Modify the uploaded file** (`confidential.txt`) on your local system.
2. Observe how quickly **pCloud synchronizes** the changes across devices.
3. **Check file version history** in pCloud to see older versions of the file (important for rollback & data recovery).
4. **Test file expiration** by setting an expiry date for a shared link and verify if access is revoked automatically.

Step 3: Advanced Persistent Threat (APT) Simulation

1. **Use an old password** and attempt multiple failed logins → pCloud will **lock the account temporarily** after multiple incorrect attempts (prevention of brute-force attacks).
2. **Enable Two-Factor Authentication (2FA)** under settings.
3. Try logging in from an **unrecognized device** → pCloud will **prompt for extra verification**.
4. Observe the **login activity logs** in pCloud for signs of suspicious access attempts.

Step 4: Client-Server Encryption Analysis

1. **Upload a file** to **pCloud Crypto (if available)** and verify that the file is **encrypted locally before upload**.
2. **Download Wireshark (optional)** and monitor **network traffic** while uploading/downloading a file.
    - If encryption is enabled, you should see **TLS-encrypted traffic** instead of plaintext data.
3. **Verify server-side encryption** by checking pCloud's security policy (data is stored in AES-256 encrypted format).

Expected Outcome:

□ **Threat Analysis:** File sharing without security can lead to data leaks, but password protection mitigates this risk.
□ **Time-Related Security:** Cloud storage offers version history and automatic expiration for shared links.
□ **APT Protection:** Security features like login activity logs, 2FA, and account lockouts prevent attacks.
□ **Client-Server Encryption:** pCloud Crypto ensures end-to-end encryption, making files unreadable on pCloud servers.

---

6. Cloud Security & Storage Analysis Using CloudMe

Objective:

To analyze **file storage, synchronization, sharing, security threats, and encryption mechanisms** using **CloudMe**, a cloud storage service that supports **client-server encryption, multi-device access, and file sharing**.

Cloud Service Used:

□ **CloudMe** (Sign up here)

- Provides **cloud storage with synchronization** across devices.
- Supports **file sharing via links**.
- Offers **client-side encryption** for enhanced security.

Software & Tools Required:

- **CloudMe Web Interface**
- **CloudMe Desktop App (for synchronization and client-side encryption)**
- **Wireshark (optional, for transport security analysis)**

Experiment Steps:

Step 1: File Storage & Synchronization

1. **Create a CloudMe account** and log in.
2. Install the **CloudMe Desktop App** on your computer.
3. Upload a **test file** (`testfile.txt`) to CloudMe via the web interface.
4. Verify that the file appears in the **CloudMe folder** on your desktop app, confirming synchronization.

Step 2: File Sharing & Security Risks

1. Right-click on `testfile.txt` in the **CloudMe web interface**.
2. Select **"Share"** and generate a **public link**.
3. Try **accessing the link** from another browser or device **without logging in**.
4. **Test security enhancement** by adding a password to the shared link and verifying unauthorized users cannot access it.

Step 3: Client-Server Encryption Analysis

1. Upload another file (`confidential.docx`) to CloudMe.
2. Open the **CloudMe desktop app settings** and enable **encryption (if available)**.
3. Check if the file is encrypted before upload using a **local encryption tool** (e.g., VeraCrypt).
4. Download the file from another device and verify that it requires decryption before opening.

Step 4: Transport Security & Threat Analysis

1. **Log in** to CloudMe from different devices (PC, mobile).
2. Check if CloudMe uses **HTTPS (SSL/TLS encryption)** by observing the browser's **lock icon** in the URL bar.
3. (Optional) Use **Wireshark** to capture network packets and verify that file transfers are encrypted in transit.
4. Attempt multiple **failed logins** to see if CloudMe **locks the account** (protection against brute-force attacks).

Expected Outcome:

□ **File Synchronization:** CloudMe ensures files are automatically updated across devices.
□ **Secure File Sharing:** Password protection prevents unauthorized access.
□ **Client-Server Encryption:** Ensures files remain private during upload/download.
□ **Transport Security:** HTTPS encryption prevents eavesdropping.
□ **Threat Protection:** CloudMe implements account security features against unauthorized access.

---

7. File Synchronization & Sharing Using Dropbox

Objective:

To demonstrate **file synchronization, backup, multi-device access, and secure file sharing** using **Dropbox**.

Software & Tools Required:

- **Dropbox Web Interface** (https://www.dropbox.com/)
- **Dropbox Desktop Application** (Download here)
- **Two devices for testing synchronization** (e.g., Laptop + Mobile Phone)

Experiment Steps:

Step 1: Dropbox Account Setup & Installation

1. **Create a Dropbox Account:**
   - Visit Dropbox Sign-Up Page.
   - Enter **name, email, and password**, then click **Sign Up**.
   - Verify your email by clicking the confirmation link sent to your inbox.
2. **Download and Install Dropbox App (Optional but Recommended):**
   - Go to Dropbox Download Page.
   - Download and install the **Dropbox Desktop Application** for your OS.
   - Sign in to your **Dropbox account** after installation.

Step 2: File Synchronization Across Devices

1. **Upload a File from the Web Interface:**
   - Log in to **Dropbox Web**.
   - Click **Upload Files** and select a file (e.g., `TestFile.txt`).
   - Wait for the upload to complete.
2. **Verify Synchronization on Another Device:**
   - Open the **Dropbox folder** on your desktop (if installed).
   - Check if the uploaded file (`TestFile.txt`) appears automatically.
   - If using a mobile device, open the **Dropbox mobile app** and check if the file is available.
3. **Modify the File on One Device & Sync the Changes:**
   - Open `TestFile.txt` from the **Dropbox folder on your PC**.
   - Add some text (e.g., "Updated on PC") and **save the file**.
   - Wait a few seconds, then check if the modified file is updated on **Dropbox Web and mobile app**.
4. **Delete a File and Restore It from Dropbox Version History:**
   - Delete `TestFile.txt` from your **Dropbox folder** on your desktop.
   - Log in to **Dropbox Web**, go to **Deleted Files**, and click **Restore**.
   - Check if the file **reappears in your Dropbox folder**.

Step 3: Secure File Sharing

1. **Generate a Shareable Link:**
   - Right-click on `TestFile.txt` in **Dropbox Web** and select **Share**.
   - Click **Copy Link** and share it with another user or test by opening it in an incognito browser.
2. **Set File Permissions for Security:**
   - In the **Share settings**, change permissions to:
     - **"Anyone with the link can view"** (default).
     - **"Invite specific people"** (only certain users can access).
     - **"Add password protection"** (Premium feature).
3. **Test File Access on a Different Device:**
   - Open the shared link on another device or browser.
   - Verify if the file is accessible **only to authorized users**.

Expected Outcome:

☐ **File synchronization works across devices in real time**.
☐ **Modified files are updated automatically**.
☐ **Deleted files can be recovered using version history**.
☐ **Secure file sharing with permission control**.

---

8. Secure File Synchronization & Backup Using TeamDrive

Objective:

To demonstrate **file synchronization, backup, multi-device access, and encryption** using **TeamDrive**.

Experiment Steps:

Step 1: TeamDrive Account Setup & Installation

1. **Create a TeamDrive Account:**
   - Visit TeamDrive Registration and sign up.
   - Verify your email and log in.
2. **Install TeamDrive Client Software:**
   - Download **TeamDrive Desktop Client** and install it.
   - Sign in with your credentials.

Step 2: File Synchronization & Backup

1. **Create a New TeamDrive Space:**
   - Open TeamDrive and create a **new space** (folder) named **"BackupTest"**.
   - Select a local folder to sync with the cloud.
2. **Upload and Sync a File:**
   - Copy a test file (e.g., `TestDoc.txt`) into the **BackupTest** folder.
   - Wait for TeamDrive to **automatically upload and sync** the file.
3. **Verify File Synchronization on Another Device:**
   - Log in to **TeamDrive Web App** on another device.
   - Check if the uploaded file appears in the **BackupTest folder**.
4. **Modify a File & Check Version Control:**
   - Edit `TestDoc.txt` (add new text) and save it.
   - Check if TeamDrive syncs the updated file.
   - Explore **Version History** to restore an older version.

Step 3: Data Recovery & Encryption Analysis

1. **Delete `TestDoc.txt`** from the **BackupTest** folder.
2. Open **TeamDrive Web App**, go to **Deleted Files**, and restore `TestDoc.txt`.
3. Check **Encryption Settings** to verify **end-to-end encryption** is enabled.

Expected Outcome:

- **Files sync automatically across devices**.
- **Version history allows restoring old file versions**.
- **Deleted files can be recovered**.
- **End-to-end encryption ensures secure file storage**.

---

9. Secure File Sharing & Access Control Using TeamDrive

Objective:

To demonstrate **secure file sharing, collaboration, and permission control** using **TeamDrive**.

Experiment Steps:

Step 1: Create a Secure Shared Folder

1. **Open TeamDrive Desktop Client or Web App.**
2. **Create a New Space (Shared Folder)** named **"ProjectDocs"**.
3. **Upload a test file** (e.g., `ProjectPlan.pdf`) to the **ProjectDocs** folder.

Step 2: Invite Users & Set Access Permissions

1. **Right-click on the "ProjectDocs" space** and select **"Invite Users"**.
2. **Enter the email address** of another TeamDrive user.
3. **Assign appropriate access permissions:**
   - **Read-Only:** User can view but not edit.
   - **Read & Write:** User can view and edit.
   - **Full Control:** User can modify, delete, and invite others.

Step 3: Test File Access Control

1. **Log in as the invited user** and check if **ProjectPlan.pdf** is accessible.
2. **Modify the file** (if permissions allow) and check if changes sync for all users.
3. **Remove a user's access** and verify they can no longer access the file.

Expected Outcome:

 Users can securely collaborate on shared files.
 Permissions control who can edit or view documents.
 Revoked users lose access to files immediately.

---

10. Simulating and Detecting MITM (Man-in-the-Middle) Attacks on Cloud Storage Traffic Using Wireshark

Objective:

To simulate a **Man-in-the-Middle (MITM) attack** on cloud storage traffic and detect it using **Wireshark**. This experiment demonstrates how an attacker can intercept **unencrypted cloud storage communication** and highlights the importance of **TLS encryption and secure authentication**.

Tools & Software Required:

 Cloud Storage Service: Google Drive, Dropbox, or OneDrive
 Wireshark: For network traffic analysis (Download here)
 MITM Attack Tools: Ettercap (Linux) or Bettercap (Kali Linux)
 Virtual Machine (Optional): Kali Linux (Attacker System)

Experiment Setup

Step 1: Configure Cloud Storage for File Transfer

1. **Create a Cloud Storage Account**
   - Sign up for **Google Drive, Dropbox, or OneDrive**.
   - Install the **desktop sync application** on your system.
2. **Upload and Download Files**
   - Upload a **sample text file (e.g., "testfile.txt")** to the cloud.
   - Download the same file to simulate **data transmission over the network**.

Step 2: Capture Cloud Storage Traffic Using Wireshark

1. **Start Wireshark on the Victim Machine**
   - Open **Wireshark** and select the **Wi-Fi or Ethernet interface**.
   - Click **Start Capture** to begin monitoring network packets.
2. **Filter Cloud Traffic**
   - Use the following **Wireshark filters** to capture cloud storage traffic:
     - **Google Drive:** `tcp.port == 443 && ip.dst_host contains "google"`
     - **Dropbox:** `tcp.port == 443 && ip.dst_host contains "dropbox"`
     - **OneDrive:** `tcp.port == 443 && ip.dst_host contains "onedrive"`
3. **Observe Encrypted (HTTPS) and Unencrypted (HTTP) Traffic**
   - If HTTPS is used, the **packets are encrypted**.
   - If unencrypted HTTP traffic is detected, **it is vulnerable to MITM attacks**.

Step 3: Simulate a MITM Attack Using Ettercap (Attacker System)

(Warning: Perform this experiment only in a controlled, ethical environment!)

1. **Launch Kali Linux (or Linux system with Ettercap installed)**
   - Open a terminal and run:

     ```
     bash
     CopyEdit
     sudo ettercap -G
     ```

   - This opens Ettercap's graphical interface.
2. **Start ARP Spoofing to Intercept Victim's Traffic**
   - Go to **Hosts > Scan for Hosts** and select the victim's IP.
   - Select **MITM > ARP Poisoning** and enable "Sniff Remote Connections."
   - Click **Start Sniffing** to intercept victim's cloud storage traffic.
3. **Monitor Intercepted Traffic in Wireshark**
   - On the attacker machine, **run Wireshark** to inspect captured packets.
   - If the victim's connection is unencrypted (HTTP), **files and credentials can be extracted**.

Step 4: Detect MITM Attack Using Wireshark on the Victim Machine

1. **Look for Duplicate ARP Replies (ARP Spoofing Detection)**
   - In Wireshark, apply the filter:

     ```
     bash
     CopyEdit
     arp.duplicate-address-detected
     ```

   - If multiple devices claim to have the same **MAC address**, an **ARP Poisoning attack** is in progress.
2. **Analyze SSL/TLS Handshake Anomalies**
   - Apply the filter:

     ```
     bash
     CopyEdit
     tls.handshake
     ```

   - If an attacker performs **SSL stripping**, TLS encryption may be downgraded to **HTTP**, making data vulnerable.
3. **Check for Suspicious DNS Responses (DNS Spoofing Detection)**
   - Apply the filter:

     ```
     bash
     CopyEdit
     dns && ip.src != your_DNS_server
     ```

   - If DNS responses originate from an unknown source, a **MITM attack is redirecting traffic**.

Expected Outcome:

□ MITM attacks can intercept unencrypted cloud storage traffic.
□ Wireshark detects ARP spoofing and SSL stripping.
□ Secure cloud storage traffic (TLS-encrypted) remains unreadable by attackers.

---

11. Testing Versioning Features in Dropbox

Objective:

To test and analyze **file versioning** in **Dropbox**, which allows users to recover previous versions of a file after modifications or deletions.

Tools & Software Required:

 Dropbox Account (Free or Paid)
 Dropbox Web Interface or Desktop App
 A Text Editor (Notepad, VS Code, or Word)

Experiment Steps

Step 1: Upload an Initial File to Dropbox

1. **Log in to Dropbox** at https://www.dropbox.com/.
2. Click on **Upload Files**, select a text file (**testfile.txt**), and upload it.
3. Verify that the file appears in your Dropbox folder.

Step 2: Modify the File Multiple Times

1. **Edit the file** on your local machine.
   - Add some text (e.g., "**Version 2 - Updated content**") and save it.
2. **Re-upload the file** to Dropbox, replacing the existing version.
3. **Repeat this process** at least **three times** to create multiple file versions.

Step 3: Access File Version History

1. **Locate the file** in your Dropbox folder.
2. **Right-click on the file** and select **Version history**.
3. A list of all previous versions appears. **Click on each version to preview or restore it.**

Step 4: Restore an Older Version

1. In the **Version history** window, choose an older version of the file.
2. Click **Restore** to replace the latest version with the selected one.
3. Open the restored file and verify that it contains the older content.

Step 5: Test Deleted File Recovery

1. **Delete the file** from Dropbox.
2. Click on the **Deleted files** section in Dropbox.
3. Find the deleted file and click **Restore** to recover it.

Expected Outcome:

 Dropbox stores multiple versions of a file and allows users to **restore previous versions**.
 Users can recover **deleted files** within the **retention period (30 days for free users, 180 days for paid users).**

---

12. Measuring Speed and Efficiency of Cloud Storage Services Using JMeter

Objective:

To evaluate the **upload speed, download speed, response time, and efficiency** of different **cloud storage services** (e.g., Dropbox, Google Drive, OneDrive) using **Apache JMeter**.

Tools & Software Required:

- ☐ **Apache JMeter** (Download Here)
- ☐ **Cloud Storage Accounts** (Dropbox, Google Drive, OneDrive, etc.)
- ☐ **Test Files (10MB, 50MB, 100MB files for performance analysis)**
- ☐ **Wireshark or SpeedTest.net (Optional for Network Benchmarking)**

Experiment Steps

Step 1: Install and Set Up JMeter

1. **Download Apache JMeter** from JMeter Official Website.
2. Extract the JMeter folder and launch:
   - **Windows:** `JMeter.bat`
   - **Mac/Linux:** `JMeter.sh`

Step 2: Create a JMeter Test Plan

1. **Open JMeter and Create a New Test Plan**
   - Click **File → New**
   - Rename it to **Cloud Storage Performance Test**
2. **Add a Thread Group** (Simulating User Requests)
   - Right-click on **Test Plan** → Add → Threads (Users) → **Thread Group**
   - Set the parameters:
     - **Number of Threads (Users):** `10` (Simulating 10 users accessing storage)
     - **Ramp-up Period:** `5` seconds
     - **Loop Count:** `5` (Each user performs 5 iterations)

Step 3: Configure HTTP Requests for Cloud Storage Performance Testing

(A) Upload Speed Test

1. Right-click on **Thread Group** → Add → Sampler → **HTTP Request**.
2. Configure the settings:

    - **Server Name/IP:** `www.dropbox.com` (for Dropbox) or respective cloud storage domain.
    - **Method:** `POST`
    - **Path:** `/upload` (depending on API endpoint of the selected service).
    - **Use Body Data**: Attach a **10MB test file** for upload.
3. Add a **Response Assertion** to verify upload success.

(B) Download Speed Test

1. Add another **HTTP Request** Sampler.
2. Configure:
    - **Method:** `GET`
    - **Path:** `/download` (specific file path in the cloud storage).
3. Use **Save Responses to a File Listener** to store downloaded files.

(C) Response Time & Performance Metrics

1. Add **View Results in Table** to track response time per request.
2. Add **Graph Results** to visualize **latency, throughput, and download/upload speeds**.

Step 4: Run the JMeter Test

1. Click **Start** in JMeter.
2. Observe results in:
    - **View Results in Table** (Response times per request).
    - **Graph Results** (Upload/download trends).
    - **Summary Report** (Overall efficiency and performance metrics).

Step 5: Analyze the Performance Metrics

 Upload Time: Measures time taken to upload files to cloud storage.
 Download Time: Measures time taken to retrieve files from cloud storage.
 Latency: Time delay before the server responds to a request.
 Throughput: Number of successful transactions per second.

---

13. Recovering Deleted or Lost OneDrive Data Using Recuva

Objective:

To demonstrate how to recover **permanently deleted or lost OneDrive files** using **Recuva**, a file recovery tool, when OneDrive is synced to a local computer.

Prerequisites:

- **A Microsoft OneDrive account**
- **OneDrive desktop application installed & synced**
- **Recuva software** (Download from Piriform)
- **Windows PC with local OneDrive folder**

Experiment Steps

Step 1: Setup OneDrive & Verify Sync

1. **Ensure OneDrive is installed & logged in** on your PC.
2. **Check the local OneDrive folder** (default location: `C:\Users\YourName\OneDrive`).
3. **Upload test files** (e.g., `testfile.docx`, `testimage.jpg`) to OneDrive and confirm they are synced online.

Step 2: Delete Files for Recovery Testing

1. **Delete a test file** (`testfile.docx`) from the **OneDrive folder**.
2. **Remove it permanently** (Shift + Delete) instead of sending it to the Recycle Bin.
3. **Verify deletion** by checking online at OneDrive.com—the file should be removed.

Step 3: Install and Run Recuva

1. **Download & Install Recuva** from Piriform.
2. **Open Recuva** and select **File Recovery Mode**.

Step 4: Scan for Deleted OneDrive Files

1. **Select file type** (Documents, Images, or All Files).
2. **Choose the location:**
   - Select the **OneDrive folder**:

```
bash
CopyEdit
C:\Users\YourName\OneDrive
```

3. Click **Start Scan** and wait for the results.
4. **Analyze scan results:**
   - **Green files** → Fully recoverable.
   - **Yellow files** → Partially recoverable (corrupted).
   - **Red files** → Unrecoverable.
5. Select the file and click **Recover**.
6. **Save the recovered file** to a different location (not OneDrive).

Step 5: Verify OneDrive Sync & Restore

1. Move the recovered file back to the **OneDrive folder**.
2. Check if **OneDrive re-uploads the file automatically**.
3. **Test OneDrive's built-in version history feature:**
   - Right-click the file in OneDrive → **View version history**.
   - Restore an **older version** if available.

Expected Outcome:

☐ Recuva should successfully recover **recently deleted OneDrive files** from local storage.
☐ If **version history** is enabled, older versions can be restored directly from OneDrive.
☐ If a file is **overwritten**, only partial recovery may be possible.

---

14. Implementing and Testing Encryption Before Uploading to Dropbox Using VeraCrypt

Objective:

To encrypt files before uploading them to **Dropbox** using **VeraCrypt**, ensuring **data security and confidentiality**, even if Dropbox storage is compromised.

Prerequisites

□ **VeraCrypt installed** (Download here)
□ **Dropbox installed and synced on your PC**
□ **Basic knowledge of encryption and cloud storage**

Experiment Steps

Step 1: Install VeraCrypt and Dropbox

1. Download and install **VeraCrypt** from its official website.
2. Install and set up **Dropbox** on your PC. Ensure **Dropbox Sync** is enabled.

Step 2: Create an Encrypted File Container in Dropbox

1. Open **VeraCrypt** and click **Create Volume**.
2. Select **Create an encrypted file container** → Click **Next**.
3. Choose **Standard VeraCrypt volume** → Click **Next**.
4. Click **Select File**, navigate to your **Dropbox folder**, and name your encrypted container (e.g., `SecureData.tc`).
5. Click **Next**.

Step 3: Configure Encryption Settings

1. Choose an **encryption algorithm** (**AES** recommended) and **hash algorithm** (**SHA-512** recommended).
2. Click **Next**.
3. Specify the **volume size** (e.g., **500MB - 1GB** for testing).
4. Set a **strong password** and click **Next**.
5. Select the **Filesystem** as **NTFS** or **exFAT**, then click **Format**.

Step 4: Mount and Use the Encrypted Volume

1. Click **Select File**, locate `SecureData.tc` in the **Dropbox folder**, and click **Mount**.
2. Enter the **password**, and the encrypted volume will be mounted as a **virtual drive** (e.g., `E:`).
3. Copy or move **sensitive files** into this virtual drive.

Step 5: Upload Encrypted Data to Dropbox

1. Once files are added, click **Dismount** in VeraCrypt.
2. **Dropbox will now sync only the encrypted file (`SecureData.tc`)**, not individual unencrypted files.
3. Check in the **Dropbox web interface** to confirm the encrypted file is uploaded.

Testing and Verification

1. Verify Data Security in Dropbox

- Try opening the uploaded `SecureData.tc` in **Dropbox Web** → It should be unreadable.

2. Test Data Recovery on Another Device

1. On another computer, download **VeraCrypt** and install **Dropbox**.
2. Download `SecureData.tc` from **Dropbox Web** or let it sync via the **Dropbox desktop app**.
3. Open VeraCrypt, select `SecureData.tc`, and click **Mount**.
4. Enter the **password** → Verify that the files are accessible.

Expected Outcome:

□ Files are **securely encrypted** before Dropbox upload.
□ Even if Dropbox is **hacked, stolen, or compromised**, the encrypted data remains **unreadable**.
□ Encrypted files can be **securely downloaded and decrypted** on any device with **VeraCrypt** and the correct **password**.

---

15. Implementing and Testing End-to-End Encryption in Cloud Storage using Cryptomator

Objective:

To encrypt files before uploading them to a cloud storage service using Cryptomator and verify the encryption effectiveness.

Prerequisites:

 Windows PC with Admin Privileges
 Cryptomator Installed (Download here: https://cryptomator.org/)
 A cloud storage account (e.g., Google Drive, OneDrive, Dropbox, etc.)
 Some test files to encrypt and upload

Experiment Steps

Step 1: Download and Install Cryptomator

1. Go to the official Cryptomator website: https://cryptomator.org/.
2. Download the Windows version of Cryptomator.
3. Run the installer and follow the on-screen instructions to complete the installation.
4. Open Cryptomator after installation.

Step 2: Create an Encrypted Vault

1. Click **+ Add Vault** in the Cryptomator interface.
2. Select **Create New Vault** and choose a name for the vault (e.g., "SecureFiles").
3. Choose a folder location where the encrypted vault will be stored.
   - This should be inside your cloud storage folder (e.g., Google Drive, Dropbox, or OneDrive).
4. Set a strong password for the vault (this password will be required to decrypt files).
5. Click **Create Vault**, then **Unlock Vault** to mount it as a virtual drive.

Step 3: Encrypt Files Using Cryptomator

1. After unlocking the vault, it will appear as a virtual drive in Windows Explorer.
2. Copy or move files into the Cryptomator virtual drive.
3. Close the Cryptomator window or click **Lock Vault** to encrypt the contents.
4. The files will now be securely encrypted within the cloud storage folder.

Step 4: Upload the Encrypted Vault to Cloud Storage

1. Ensure your cloud storage sync client (Google Drive, Dropbox, OneDrive) is running.
2. The encrypted Cryptomator vault will be automatically uploaded to the cloud.
3. Check your cloud storage to confirm the encrypted files are stored.

Step 5: Verify Encryption Effectiveness

1. Attempt to open the encrypted vault files directly from cloud storage (they should be unreadable).
2. Try accessing the Cryptomator vault with an incorrect password (it should deny access).
3. Unlock the vault using Cryptomator and check if the original files are accessible.

Testing and Verification

1. Detecting Encryption Security

☐ Check if files inside the vault appear as unreadable when accessed from cloud storage.
☐ Ensure that the vault does not reveal original filenames or file contents.
☐ Confirm that only the correct Cryptomator password can decrypt the files.

2. User Awareness & Security Best Practices

☐ Use a strong, unique password for the Cryptomator vault.
☐ Backup the vault password securely (Cryptomator does not provide password recovery).
☐ Regularly update Cryptomator and your cloud storage security settings.

Expected Outcome:

☐ Files stored in the cloud are securely encrypted and unreadable without the correct password.
☐ Users can protect sensitive data from unauthorized access.
☐ The encryption process ensures data privacy even if cloud storage is compromised.

16. Automating Cloud Backup, Synchronization, and File Transfer Using Rclone

Objective:

To automate cloud backup, synchronization, and file transfer between different cloud storage services using Rclone.

Prerequisites:

 Windows PC with Admin Privileges
 Rclone Installed (Download here: https://rclone.org/downloads/)
 At least two cloud storage accounts (e.g., Google Drive, OneDrive, Dropbox, etc.)
 Some test files for backup and synchronization

Experiment Steps

Step 1: Download and Install Rclone

1. Go to the official Rclone website: https://rclone.org/downloads/.
2. Download the Windows version of Rclone.
3. Extract the downloaded ZIP file to a preferred location.
4. Open a Command Prompt (cmd) and navigate to the extracted folder.

Step 2: Configure Cloud Storage Accounts in Rclone

1. Open a Command Prompt and type:

   ```
   rclone config
   ```

2. Select **'n'** to create a new remote storage connection.
3. Enter a name for the cloud storage (e.g., `mydrive` for Google Drive).
4. Choose the cloud storage type (e.g., `drive` for Google Drive, `onedrive` for OneDrive, etc.).
5. Follow the prompts to authenticate and authorize Rclone access.
6. Repeat the process for a second cloud storage account if needed.

Step 3: Test Cloud Storage Connection

1. Verify configured accounts by running:

   ```
   rclone listremotes
   ```

2. Check files in a cloud storage account:

   ```
   rclone ls mydrive:
   ```

Step 4: Automate Cloud Backup Using Rclone

1. To copy local files to cloud storage, run:

   ```
   rclone copy C:\local_folder mydrive:/backup_folder
   ```

2. To synchronize local files with cloud storage, run:

   ```
   rclone sync C:\local_folder mydrive:/sync_folder
   ```

3. To automate backups, create a batch file (`backup.bat`) with the command:
4. `@echo off`
5. `rclone copy C:\local_folder mydrive:/backup_folder`
   `exit`

6. Schedule the batch file to run automatically using Windows Task Scheduler.

Step 5: Transfer Files Between Cloud Storage Services

1. To transfer files from Google Drive to OneDrive:

   ```
   rclone copy mydrive:/backup_folder myonedrive:/backup_folder
   ```

2. To synchronize two cloud storage accounts:

   ```
   rclone sync mydrive:/sync_folder myonedrive:/sync_folder
   ```

Testing and Verification

1. Checking Backup Integrity

 Compare file size and count in both source and destination.
 Run `rclone check` to verify if files match:

```
rclone check mydrive:/backup_folder myonedrive:/backup_folder
```

2. Ensuring Automation Works

 Verify Windows Task Scheduler logs to ensure scheduled backups run successfully.
 Test synchronization by modifying local files and checking if they update in cloud storage.

Expected Outcome:

- Automated file backup to cloud storage without manual intervention.
- Secure and synchronized files across multiple cloud storage services.
- Efficient cloud file transfers and backup management using Rclone.

---

17. Monitoring and Analyzing Cloud Storage Traffic Using Wireshark

Objective:

To monitor and analyze cloud storage traffic using Wireshark on Windows, identifying security risks and traffic patterns.

Prerequisites:

 Windows PC with Admin Privileges
 Wireshark Installed (Download here: https://www.wireshark.org/download.html)
 A cloud storage account (e.g., Google Drive, OneDrive, Dropbox)
 Some test files for upload and download

Experiment Steps

Step 1: Download and Install Wireshark

1. Go to the official Wireshark website: https://www.wireshark.org/download.html.
2. Download the Windows installer and run the setup.
3. Follow the installation instructions and install the necessary network drivers (Npcap).
4. Launch Wireshark after installation.

Step 2: Select the Network Interface

1. Open Wireshark and go to the **Capture** section.
2. Select the network interface connected to the internet (e.g., Wi-Fi or Ethernet).
3. Click the **Start Capture** button.

Step 3: Capture Cloud Storage Traffic

1. Open a cloud storage service (Google Drive, Dropbox, or OneDrive) in a web browser or desktop application.
2. Upload a test file to the cloud storage.
3. Download the same file from the cloud storage.
4. While these actions take place, Wireshark captures the network packets.
5. Click the **Stop Capture** button once the file transfer is complete.

Step 4: Analyze the Captured Traffic

1. Use the **Filter Bar** in Wireshark to focus on cloud storage traffic:
   - **Google Drive:** `ip.addr == 172.217.0.0/16`
   - **Dropbox:** `ip.addr == 162.125.0.0/16`
   - **OneDrive:** `ip.addr == 13.107.0.0/16`
2. Apply **HTTP and HTTPS** filters:

   ```
   http || tls
   ```

3. Identify key packet information:
   - Source and Destination IP Addresses
   - Protocols (TLS, HTTP, TCP)
   - Packet size and timing

Step 5: Detect Security Issues

1. Check for **unencrypted traffic** (HTTP instead of HTTPS).
2. Identify potential **data leaks** or **unexpected connections**.
3. Analyze traffic behavior for abnormal data transfers.

Testing and Verification

1. Identifying Cloud Storage Traffic Patterns

☐ Observe the number of packets sent during upload vs. download.
☐ Identify cloud service IPs from captured packets.

2. Detecting Unsecured Data Transfers

☐ Verify that traffic is encrypted using **TLS protocol**.
☐ Look for any **HTTP** traffic, which may indicate security risks.

3. Ensuring Proper Network Security Measures

☐ Check if Multi-Factor Authentication (MFA) is in use.
☐ Analyze logs for unauthorized access attempts.

Expected Outcome:

☐ Users can monitor cloud storage traffic effectively.
☐ Security vulnerabilities, such as unencrypted data transmission, can be identified.
☐ Understanding of cloud storage data flow and encryption techniques is improved.

---

18. Conducting a Cloud Storage Data Integrity Check Using Checksums

Objective:

To verify data integrity before and after cloud storage upload/download using checksum tools such as HashCalc.

Prerequisites:

 Windows PC with Admin Privileges
 HashCalc Installed (Download here: https://www.slavasoft.com/hashcalc/)
 A cloud storage account (e.g., Google Drive, OneDrive, Dropbox)
 A sample test file for integrity verification

Experiment Steps

Step 1: Download and Install HashCalc

1. Go to the official HashCalc website: https://www.slavasoft.com/hashcalc/.
2. Download the Windows installer and run the setup.
3. Follow the installation instructions and complete the setup.
4. Launch HashCalc after installation.

Step 2: Generate Checksum for Original File

1. Open HashCalc.
2. Click the **Browse** button and select the sample test file.
3. Choose a hashing algorithm (e.g., **MD5, SHA-256, SHA-512**).
4. Click **Calculate** to generate the checksum.
5. Note down the generated checksum for future comparison.

Step 3: Upload File to Cloud Storage

1. Log in to your cloud storage account (Google Drive, OneDrive, Dropbox, etc.).
2. Upload the test file to the cloud.
3. Ensure the upload completes successfully.

Step 4: Download File from Cloud Storage

1. Download the uploaded file from the cloud storage to your local system.

2. Save it in a different location than the original file.

Step 5: Generate Checksum for Downloaded File

1. Open HashCalc again.
2. Click the **Browse** button and select the downloaded file.
3. Choose the same hashing algorithm used earlier.
4. Click **Calculate** to generate the checksum.

Step 6: Compare Checksums

1. Compare the checksums of the original and downloaded files.
2. If the checksums match, the file has not been altered.
3. If the checksums differ, the file has been modified or corrupted during upload/download.

Testing and Verification

1. Ensuring Data Integrity

 If both checksums match, data integrity is maintained.
 If checksums do not match, investigate possible corruption or tampering.

2. Testing with Different File Types

 Repeat the process with text files, images, videos, and compressed files.
 Observe any variations in checksum results.

3. Checking for Potential Data Corruption

 Try uploading/downloading files from different networks.
 Verify if checksum differences occur in specific conditions.

Expected Outcome:

 Users can verify the integrity of files before and after cloud storage operations.
 Corrupt or altered files can be detected using checksum verification.
 Enhanced awareness of data security and integrity maintenance in cloud storage.

---

19. Configuring and Testing Role-Based Access Control (RBAC) in Google Drive

Objective:

To configure and test Role-Based Access Control (RBAC) and permission settings in Google Drive to ensure secure file access and management.

Prerequisites:

 Windows PC with Admin Privileges
 A Google Drive account
 A test folder and sample files for access control
 Multiple Google accounts (or the ability to add users with different roles)

Experiment Steps

Step 1: Create a Folder for Testing RBAC

1. Log in to your **Google Drive** account.
2. Click on **New** → **Folder** and name it **RBAC_Test_Folder**.
3. Upload a few sample files to the folder.

Step 2: Identify Available Roles in Google Drive

1. Google Drive offers different sharing roles:
   - **Viewer**: Can only view files
   - **Commenter**: Can view and comment on files
   - **Editor**: Can view, edit, and delete files
   - **Owner**: Full control, including managing roles

Step 3: Assign Roles and Permissions

1. Right-click the **RBAC_Test_Folder** and select **Share**.
2. Enter test user email addresses.
3. Assign different roles to different users:
   - User A → **Viewer**
   - User B → **Editor**
   - User C → **Owner** (if applicable)
4. Click **Send** to share the folder.

Step 4: Test Role-Based Access Control

1. Log in as **User A (Viewer)** and attempt the following:
   - Open files (□ Success)
   - Edit files (□ Should be restricted)
   - Delete files (□ Should be restricted)
2. Log in as **User B (Editor)** and attempt:
   - Open files (□ Success)
   - Edit files (□ Success)
   - Delete files (□ Success)
3. Log in as **User C (Owner)** and attempt:
   - Open files (□ Success)
   - Edit files (□ Success)
   - Delete files (□ Success)
   - Change permissions (□ Success)

Step 5: Modify and Revoke Permissions

1. As the **Owner**, modify User B's role from **Editor** to **Viewer**.
2. Verify that User B can no longer edit files.
3. Remove User A's access and confirm that they cannot open the folder anymore.

Testing and Verification

1. Verifying Access Restrictions

□ Ensure that each user can only perform actions allowed by their assigned role.
□ Try to access the folder from an unauthorized account (should be denied).

2. Checking Permission Changes in Real-Time

□ Modify permissions and check if they apply immediately.
□ Ensure revocation works as expected (User loses access after being removed).

Expected Outcome:

□ Users only have access based on assigned roles.
□ Unauthorized actions are blocked as per RBAC settings.
□ Role changes and revocations take effect immediately.
□ Enhanced security by implementing fine-grained access controls.

20. Setting Up Duplicati for Automated Encrypted Cloud Backups and Secure Data Restoration

Objective:

To configure Duplicati for automatic encrypted cloud backups and test secure data restoration.

Prerequisites:

 Windows PC with Admin Privileges
 Duplicati Installed (Download here)
 A cloud storage account (Google Drive, Dropbox, OneDrive, etc.)
 Encryption passphrase for securing backups

Experiment Steps

Step 1: Install Duplicati

1. Download Duplicati from the official website: https://www.duplicati.com/download
2. Run the installer and follow the on-screen instructions.
3. After installation, open **Duplicati** from the Start Menu or System Tray.

Step 2: Configure a New Backup Job

1. Click **Add Backup** → Select **Configure a new backup** → Click **Next**.
2. **Backup Name:** Enter a name for the backup job.
3. **Encryption:** Choose **AES-256 encryption** and set a strong passphrase.
4. Click **Next**.

Step 3: Select Backup Storage Destination

1. Choose a cloud storage provider (Google Drive, Dropbox, OneDrive, etc.).
2. Authenticate Duplicati with the chosen cloud service.
3. Select or create a folder to store backups.
4. Click **Test Connection** to verify access.

Step 4: Select Files and Folders to Backup

1. Choose important folders for backup (e.g., Documents, Pictures, etc.).

2. Exclude unnecessary files if needed.
3. Click **Next**.

Step 5: Set Backup Schedule and Retention Policy

1. Configure the backup frequency (Daily, Weekly, Custom Schedule).
2. Set retention policy (Keep all backups, Smart Retention, or Custom rules).
3. Click **Next**.

Step 6: Run and Monitor the Backup

1. Click **Save and Run Backup**.
2. Monitor progress in the **Home Dashboard**.
3. Verify completion and check logs for errors.

Step 7: Test Data Restoration

1. Click **Restore** → Select the backup destination.
2. Choose a backup version and select files to restore.
3. Specify the restore location (Original or Custom folder).
4. Click **Restore** and verify the integrity of restored files.

Testing and Verification

1. Verifying Backup Success

☐ Check backup logs for completion status and errors.
☐ Confirm encrypted files in the cloud storage folder.

2. Ensuring Data Security

☐ Verify that cloud backups cannot be accessed without the encryption passphrase.
☐ Ensure backup integrity by restoring and comparing files.

3. Checking Automated Backup Execution

☐ Allow Duplicati to run scheduled backups.
☐ Verify periodic logs to ensure consistency.

Expected Outcome:

- Automatic encrypted backups are stored securely in the cloud.
- Data can be restored successfully with the correct passphrase.
- Backup automation runs on schedule without manual intervention.
- Enhanced data security through encryption and cloud storage.

21. Securely Uploading, Sharing, and Managing Files in TeraBox Cloud

Objective: Learn how to upload, organize, share, and manage files securely using TeraBox Cloud on Windows.

Prerequisites:

- Windows PC with an internet connection
- TeraBox account (Sign up here)
- TeraBox desktop app or web interface

Experiment Steps

Step 1: Install and Set Up TeraBox

1. **Download** and install the **TeraBox** desktop app from the official website.
2. **Sign in** to your TeraBox account or **create a new account** if needed.

Step 2: Upload Files to TeraBox

1. Open the **TeraBox desktop app** or log in via the **web interface**.
2. Click on the **Upload** button.
3. Select a **file or folder** from your computer.
4. Wait for the upload process to complete.

Step 3: Organizing Files

1. Click on **New Folder** to create a folder.
2. Drag and drop files into different folders to organize them.
3. Use the **rename, delete, and move** options to manage files efficiently.

Step 4: File Sharing and Permissions

1. Select a file or folder you want to share.
2. Click on the **Share** option.
3. Choose **Public Link** or **Password-Protected Link**.
4. Copy and send the link to others.

Step 5: Downloading and Accessing Files Offline

1. Open TeraBox and navigate to your uploaded files.
2. Click **Download** to save files back to your local device.
3. Check if the downloaded file matches the original to ensure data integrity.

Testing and Verification

☐ Verify if files are successfully uploaded and accessible from another device.
☐ Test file sharing by opening the shared link in an **incognito browser window**.
☐ Try downloading a file and compare its **size and integrity** with the original.

Expected Outcome:

☐ Successfully uploaded, organized, and shared files on **TeraBox Cloud**.
☐ Verified file accessibility and download functionality.
☐ Ensured data integrity of uploaded and downloaded files.

22. Managing Files and Synchronization Using OpenDrive Cloud

Objective: Learn how to upload, synchronize, and manage files securely using **OpenDrive Cloud** on Windows.

Prerequisites:

 Windows PC with an internet connection
 OpenDrive account (Sign up here)
 OpenDrive desktop application or web interface

Experiment Steps

Step 1: Install and Set Up OpenDrive

1. **Download** the OpenDrive desktop app from the official website.
2. Install the software and **sign in** with your OpenDrive account.
3. You will see a new **OpenDrive virtual drive** in **File Explorer**.

Step 2: Upload Files to OpenDrive

1. Open the **OpenDrive web interface** or navigate to the **OpenDrive virtual drive** in **File Explorer**.
2. Drag and drop a **file or folder** into OpenDrive.
3. Wait for the upload process to complete.
4. Verify that the file is visible in the OpenDrive web interface.

Step 3: Enabling File Synchronization

1. Open the OpenDrive **desktop application**.
2. Navigate to **Settings > Sync & Backup**.
3. Choose a **local folder** to sync with OpenDrive.
4. Enable **real-time synchronization** and save settings.
5. Test by adding a file to the local folder and checking if it appears in OpenDrive.

Step 4: File Sharing and Access Control

1. Select a file or folder in OpenDrive.
2. Click on the **Share** option.
3. Choose between **Public Link** or **Private Access with Permissions**.
4. Copy the link and test it in another browser.

5. Set an **expiration date or password** for additional security.

Step 5: Downloading and Restoring Files

1. Open the OpenDrive web interface or desktop app.
2. Select a previously uploaded file.
3. Click **Download** and save the file to your computer.
4. Compare the file size and integrity with the original.

Testing and Verification

□ Check if the uploaded file appears in both the **OpenDrive web and desktop app**.
□ Modify a synced file locally and verify that it updates automatically in OpenDrive.
□ Open a shared link in an **incognito window** and check accessibility.
□ Download a file and confirm that it is identical to the original.

Expected Outcome:

□ Successfully **uploaded, synchronized, and managed files** in OpenDrive.
□ **Verified file sharing and access control** features.
□ Ensured **data integrity and real-time synchronization**.

23. Configuring Two-Factor Authentication (2FA) in TeamDrive to Enhance Security

Objective: Learn how to enable and test **Two-Factor Authentication (2FA)** in **TeamDrive** for enhanced cloud storage security.

Prerequisites:

□ Windows PC with **TeamDrive** installed (Download here)
□ **TeamDrive account**
□ **Authenticator app** (e.g., Google Authenticator, Microsoft Authenticator, or Authy) installed on a mobile device

---

Experiment Steps

Step 1: Install and Set Up TeamDrive

1. **Download and install** the TeamDrive client on your Windows PC.
2. Launch **TeamDrive** and **sign in** with your account.
3. If you don't have an account, create one using your email.

Step 2: Enable Two-Factor Authentication (2FA)

1. **Open TeamDrive settings** by clicking on your profile or security settings.
2. Navigate to **Security > Two-Factor Authentication**.
3. Click on **Enable 2FA**.
4. A **QR code** or **setup key** will be displayed.

Step 3: Configure 2FA Using an Authenticator App

1. Open **Google Authenticator** (or your chosen authenticator app) on your mobile device.
2. Tap **Add Account > Scan QR Code** (or enter the setup key manually).
3. Your authenticator app will generate a **6-digit verification code**.
4. Enter the code in TeamDrive to verify and complete the setup.

Step 4: Testing 2FA for Secure Login

1. **Sign out** from TeamDrive.
2. Attempt to **log in again** with your credentials.
3. After entering your password, you will be prompted for a **2FA code**.

4. Open your authenticator app, enter the code, and verify successful login.

Step 5: Backup and Recovery Options

1. Go to the **2FA settings** in TeamDrive.
2. **Save backup codes** securely in case you lose access to your authenticator app.
3. Test the **backup codes** by using one instead of the 2FA code.

Testing and Verification

□ **Login Security:** Ensure that logging in now requires both a password and an authenticator-generated code.
□ **Backup Code Test:** Verify that backup codes work if you cannot access your mobile authenticator.
□ **Unauthorized Access Prevention:** Try logging in from another device without 2FA, ensuring access is denied.

Expected Outcome:

□ Successfully enabled and tested **Two-Factor Authentication (2FA)** in **TeamDrive**.
□ Enhanced cloud storage security by adding an extra layer of authentication.
□ Verified **backup codes and alternative login methods**.

24. Testing Secure File Deletion from Dropbox Cloud Storage Using BleachBit

Objective: Learn how to securely delete files from **Dropbox** cloud storage using **BleachBit**, ensuring data cannot be recovered.

Prerequisites:

□ Windows PC with **Dropbox installed** (Download here)
□ **Dropbox account** with some uploaded test files
□ **BleachBit tool** installed (Download here)

Experiment Steps

Step 1: Upload Test Files to Dropbox

1. Open **Dropbox** (web or desktop client).
2. Upload a test file (e.g., **TestFile.txt**).
3. Ensure the file syncs with the cloud by checking Dropbox online.

Step 2: Delete the File from Dropbox Normally

1. Locate **TestFile.txt** in Dropbox.
2. Right-click and select **Delete**.
3. Go to **Dropbox Trash** and delete the file permanently.

□ **Issue:** Regular deletion only removes the reference to the file; data may still be recoverable.

Step 3: Secure File Deletion Using BleachBit

1. Open **BleachBit** on your Windows PC.
2. Click **File > Shred Files**.
3. Select your **Dropbox folder** (`C:\Users\YourName\Dropbox`).
4. Choose the deleted file and click **Shred**.
5. BleachBit will overwrite the file's disk space, making recovery impossible.

Step 4: Verify Secure Deletion

1. Use **file recovery software** (e.g., Recuva) to check if the file can be recovered.
2. If properly shredded, the file should **not appear** in the recovery scan.

Testing and Verification

- **Check Dropbox Trash:** Ensure the file is permanently deleted.
- **Run a file recovery scan:** Verify that the file is irretrievable.
- **Check Dropbox sync:** Ensure secure deletion does not affect other files.

Expected Outcome:

- Successfully **shredded and securely deleted files** from **Dropbox**.
- **Verified that deleted files are non-recoverable**.
- Ensured **Dropbox continues normal operation** after secure deletion.

25. Uploading, Downloading, and Sharing Files Using Icedrive

Objective:
To upload, download, and securely share files using **Icedrive**, a cloud storage service known for its built-in encryption and modern interface.

Prerequisites:

 Windows PC with an internet connection
 Icedrive account (Sign up here)
 Icedrive Web Interface (No software installation required)

Experiment Steps

Step 1: Create an Icedrive Account and Log In

1. Go to Icedrive's official website.
2. Click **Sign Up**, enter your **email and password**, and create an account.
3. Log in to **Icedrive Web** to access your cloud storage.

Step 2: Upload a File to Icedrive

1. Click the **Upload** button.
2. Select a **file from your computer** (e.g., `document.pdf`).
3. Click **Open** and wait for the file to be uploaded.

Step 3: Download a File from Icedrive

1. Locate the uploaded file in Icedrive.
2. Click on the file and select **Download**.
3. Open the downloaded file to verify its integrity.

Step 4: Share a File Securely

1. Right-click on the file and select **Share**.
2. Copy the **generated link** and share it with others.
3. (Optional) Enable **password protection** for secure file sharing.

Testing and Verification

✔ Successfully **uploaded** a file to Icedrive.
✔ Verified **file download and integrity**.
✔ Tested **file sharing via secure link**.

Expected Outcome:

□ Users can store, download, and share files using **Icedrive**.
□ Files remain **accessible and secure** from any device.
□ Users understand **basic cloud storage operations**.