International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

A SECURITY ANALYSIS OF WEBSITE-ENABLED DIRECT FILE UPLOADS TO CLOUD STORAGE SERVICES

S SABIYA SULTHANA [1], BANDI SANJUNA [2], VINJAM VANISRI [3], BANDA NITHIN REDDY [4], BONAGIRI NIKHIL [5]

[1]Assistant Professor, Dept. of CSE, Guru Nanak Institutions Technical Campus, Ibrahimpatnam, Telangana, India

[2, 3, 4, 5] B.Tech, Dept. of CSE, Guru Nanak Institutions Technical Campus, Ibrahimpatnam, Telangana, India

ABSTRACT— With the increasing reliance on cloud storage services for handling large volumes of user data, websites have begun enabling direct file uploads from users to cloud platforms. While this approach offers greater convenience and scalability, it also introduces new security challenges due to the involvement of multiple entities, including web users, web servers, and cloud storage providers. In this study, we present the first comprehensive security evaluation of this direct upload model. Through an in-depth investigation, we identify six distinct categories of vulnerabilities and perform large-scale testing across the top 500 websites ranked by Alexa. Our findings reveal that 182 websites (36.4%) utilize cloud storage services, and a focused analysis of 28 popular websites with upload functionality shows that all exhibit at least one of the identified vulnerabilities. In total, we uncover 79 previously unreported vulnerabilities, which we responsibly disclosed to the respective platforms, including major services like Google, Reddit, and CSDN. The positive responses highlight the practical impact of our findings. We further examine the root causes of these issues and suggest effective mitigation strategies. This work contributes valuable insights into the security implications of cloud-based file uploads and aims to guide both developers and researchers in building more secure web applications.

Index Terms – Cloud storage security, direct file upload, web application security, cloud computing, vulnerability analysis, secure file upload, cloud storage services, web security, data protection, security assessment, vulnerability disclosure, secure web applications.

I. INTRODUCTION

With the ever-increasing popularity of websites, the user base and data volumes of websites increase by a large margin. As a result, storing and managing user generated massive data become a significant challenge. For example, according to statistical data, popular websites like Google and Reddit have hundreds of millions of users. Gmail alone boasts over 1.8 billion users among Google services. Reddit has approximately 430 million monthly active users. Therefore, it is important to address the aforementioned massive users' data storage and management issues. Cloud storage services have emerged as the preferred choice for website data storage and management due to their flexibility, scalability, and pay-as-you-go storage solutions. These features cater to the needs of websites across various sizes and industries .Under the cloud storage service, a new file upload scenario has been introduced, where the files are directly uploaded from web users to the cloud. Generally, there are three critical stages in this new file upload scenario: (1) upload credential requesting and dispatching, (2) file uploading and verification, and (3) callback notification and response. In stage (1), when a user wants to upload a file on a website, the client1 sends an HTTP request to the web server to ask for an upload credential. Then the web server should check the user's identity and then dispatch the upload credential if passed. In stage (2), the client uses the credential to upload the file directly to the cloud storage service. The cloud storage service should verify the credential's signature information and check whether the uploaded files satisfy the credential's policy requirements. In stage (3), the cloud storage service or the client should send a callback notification to inform the web server of the file upload results. In contrast to traditional methods where the web server stores user files locally or as an intermediary to transfer files to cloud services, this new scenario eliminates the need for the server to manage data storage or the transit of uploaded files. This offers convenience and reduces transmission overhead for web servers. However, it also introduces new attack surfaces, as follows. Firstly, the process involves authentication interactions among three roles, and each role

should be responsible for its own security authentication. For instance, if the web server or cloud storage service fails to provide proper identity authentication for a user, it opens up possibilities for attackers to forge the user's identity. Secondly, the new scenario introduces a new authorization mechanism—upload credentials. Improper setting of the upload credential or the policy may cause privilege escalation vulnerabilities. For instance, adversaries may exploit these vulnerabilities to steal or modify other users' files in cloud storage services. Finally, the new scenario involves information synchronization among the three roles. If the web server or cloud storage service does not provide a secure synchronization information verification mechanism, the authenticity of the callback notification information cannot be guaranteed. Then, attackers can forge or tamper with callback notifications to trick the web server. Previous studies on cloud storage security focus on mobile applications with the misuse of user credentials and the data management issues. However, to the best of our knowledge, no one has studied the security issues of this new scenario. There are two main challenges in analyzing the security risks of this scenario.1) It is not trivial to understand the complex interaction mechanism between different roles and determine whether it contains vulnerabilities. Many interactions are invisible to a regular web user. For instance, in practice, a user is not able to access the communication process between a website and the cloud storage. 2) It is not trivial to determine whether the concrete implementations or deployments of an interaction mechanism are vulnerable. The complexity arises due to the variety of cloud service vendors and websites, each usually having distinct implementations or deployments. In this paper, we perform the first systematic study of the security risks of directly uploading files from web users to a cloud storage service.

To address the above challenges, we conduct an in-depth analysis of the three critical stages in this scenario based on the public development manuals provided by cloud service vendors. From the perspectives of access control, credential validity, file restrictions, data integrity, data confidentiality, and consistency, we summarize the newly identified vulnerabilities into the following six types: unrestricted upload credential acquisition (V1), upload credentials validity flaw (V2), unrestricted file types and file size (V3), file overwriting (V4), file stealing (V5), and callback notification spoofing (V6). These

vulnerabilities can cause serious harm to web users, web servers, and cloud storage services, such as resource-consumption attacks, privacy leakage, sensitive data overwriting, etc.

II. RELATED WORK

A. URadar: Discovering unrestricted file upload vulnerabilities via adaptive dynamic testing

URadar is an automated security testing framework designed to discover unrestricted file upload vulnerabilities in cloud-backed web applications. It models the three-stage direct-to-cloud upload workflow—credential issuance, client-to-cloud upload, and callback synchronization—to expose gaps attackers can exploit. Using adaptive dynamic testing, URadar generates and refines targeted upload credentials and file payloads to probe real-world policy and validation weaknesses. The system simulates malicious actors who attempt privilege escalation, file overwrites, unauthorized downloads, and callback spoofing to reveal practical attack paths. URadar correlates observable behaviors across the web server, client, and cloud storage service to infer invisible interactions and detect inconsistencies. It includes modules for credential abuse detection, file-type/size bypass testing, integrity checks, and callback authenticity validation. When a potential vulnerability is found, URadar performs confirmation steps (safe, non-destructive where possible) to reduce false positives. The framework automatically produces human-readable reports that describe the vulnerability, the exploited vector, PoC steps, and recommended fixes.

B. Credit karma: Understanding security implications of exposed cloud services through automated capability inference

Credit Karma is a security analysis framework designed to understand and assess the risks associated with exposed cloud services by performing automated capability inference. The project focuses on identifying misconfigured or publicly accessible cloud components that may unintentionally expose sensitive data or functionalities. By analyzing access policies, API endpoints, and resource permissions, Credit Karma automatically infers what an attacker could potentially do if the exposed service is exploited. It simulates real-world attack scenarios to evaluate privilege escalation, unauthorized data access, and service manipulation risks. The system employs automated scanning and dynamic testing to map out exposure surfaces across multiple cloud platforms. It categorizes detected misconfigurations based

on severity and potential business impact, helping organizations prioritize remediation. Credit Karma also correlates findings across different services to detect chained vulnerabilities that may arise due to weak cross-service authentication. The framework generates comprehensive reports detailing the identified exposures, inferred attacker capabilities, and recommended mitigation strategies. Ultimately, the project aims to improve cloud security visibility, prevent data breaches, and help developers deploy secure, well-configured cloud environments.

C. Understanding and detecting abused image hosting modules as malicious services

This project, "Understanding and Detecting Abused Image Hosting Modules as Malicious Services," focuses on identifying and analyzing how legitimate image hosting platforms and modules are misused to carry out malicious activities. The study explores how attackers exploit these services to host phishing pages, malware payloads, or command-and-control content under the guise of harmless images. It investigates common abuse patterns, such as steganographic embedding of malicious code and the misuse of API-based upload functions to bypass content moderation. The system employs automated crawling, static and dynamic content inspection, and behavior-based analysis to detect suspicious uploads. By comparing normal and malicious hosting behaviors, the project builds classification models capable of recognizing abuse with high accuracy. It also examines weak authentication, lack of content validation, and insufficient URL filtering as root causes of such exploitation. Detected cases are further analyzed to understand attacker strategies, distribution techniques, and the propagation of malicious links. Finally, the project proposes a defense framework that integrates automated abuse detection with enhanced verification policies, aiming to secure image hosting platforms and protect users from hidden cyber threats.

D. Don't trust the clouds easily: The insecurity of content security policy based on object storage

The project "Don't Trust the Clouds Easily: The Insecurity of Content Security Policy Based on Object Storage" investigates the hidden security risks that arise when web applications rely on cloud object storage services to host and deliver web content. It focuses on how developers commonly configure Content Security Policies (CSP) to trust entire cloud domains, unknowingly introducing new attack surfaces. Attackers can exploit misconfigured or shared storage

buckets to inject malicious scripts, host fake resources, or perform cross-site scripting (XSS) attacks under trusted domains. The project systematically analyzes how weak or overly broad CSP rules undermine the intended protection mechanisms of modern browsers. Using large-scale empirical testing, it examines real-world websites that depend on Amazon S3, Google Cloud Storage, and other object storage platforms. The system automates the discovery of insecure CSP configurations, correlates them with exposed storage resources, and evaluates their exploitability. It further demonstrates practical attack scenarios to show how trust abuse in CSP policies can compromise user data and web integrity. Finally, the project proposes mitigation strategies—such as granular domain whitelisting, signed URL enforcement, and integrity verification—to strengthen CSP deployment in cloud-integrated environments.

III. PROPOSED METHODOLOGY

A systematic study of security risks was conducted to examine a modern cloud-based file upload scenario, where users upload files directly from websites to cloud storage services. Unlike traditional methods that route files through web servers, this direct interaction introduces multiple roles—users, web servers, and cloud services—each requiring secure coordination. The study identifies six major categories of vulnerabilities, including improper handling of upload credentials, unrestricted file types and sizes, file overwriting risks, unauthorized file access, and spoofed callback notifications. Through an extensive evaluation of the top 500 Alexa-ranked websites, it was found that 36.4% utilize cloud storage, and among 28 popular sites that allow uploads, all contained at least one vulnerability. In total, 79 new vulnerabilities were uncovered. This research is significant in highlighting the emerging threats in direct-to-cloud uploads and offers guidance on preventing these issues through improved authentication, access control, and secure communication mechanisms.

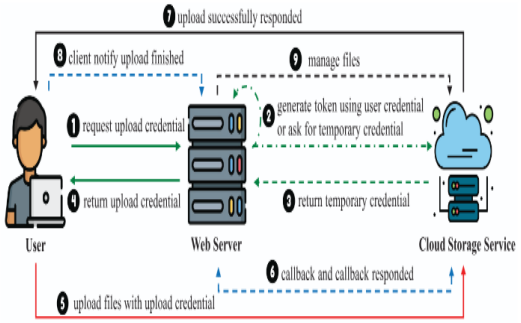
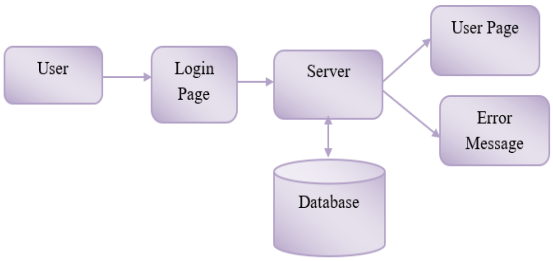


Fig. 1: System Overview

1. User Interface Design

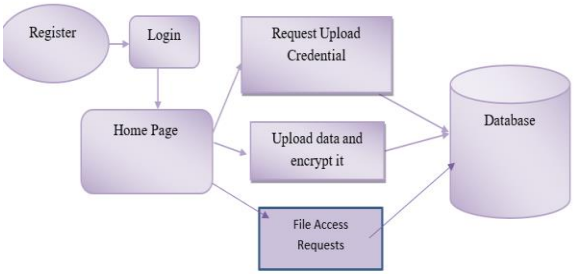
To connect with server user must give their username and password then only they can

able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password, Email id, City and Country into the server. Database will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page. It will search the query and display the query.



2. User

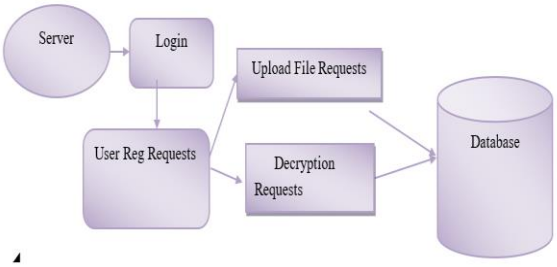
The User module serves as the client-side interface in the system, handling all interactions with the Server and cloud storage. It allows authenticated users to request upload credentials, which are required to securely upload files directly to the cloud. Once the credential is obtained, the User module encrypts the file content using RSA-based encryption (or hybrid RSA+AES for efficiency) before uploading, ensuring that the data remains confidential and tamper-proof. The module also manages download requests, either retrieving files from the cloud or requesting the corresponding decryption key from the Server to access encrypted files safely. Additionally, it performs systematic tests for the six vulnerability types (V1–V6) in a controlled and non-destructive manner, including verifying file type restrictions, preventing unauthorized access, and handling callback notifications.



3. Server

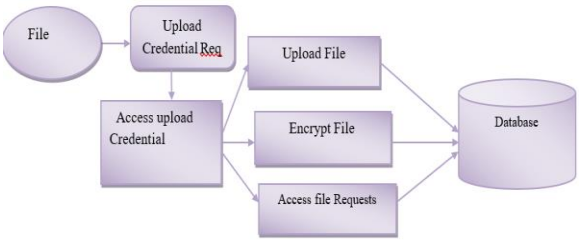
The Server module acts as the central authority that manages authentication, authorization, and key distribution. It verifies user identities, issues scoped and time-limited upload credentials, and generates RSA key pairs or encrypts AES keys for each file. The Server also validates cloud callbacks by checking signatures, nonces, and file metadata consistency to prevent spoofing or tampering. Moreover, it mediates the secure release of decryption keys only to authorized users, enforces file ownership and integrity policies, and logs all activities for auditing. Together, the User and Server modules work in coordination to provide secure file storage,

maintain confidentiality and integrity, and systematically mitigate all six vulnerability types in the direct-to-cloud upload workflow.



4. File Vulnerabilities & RSA

User module requesting an upload credential from the Server, which authenticates the user and issues a scoped, time-limited credential. The algorithm first tests for unrestricted credential acquisition (V1) and credential validity flaws (V2) by attempting controlled requests with invalid, expired, or role-switched credentials to see if the Server incorrectly issues or accepts them. Once a valid credential is obtained, the User module prepares the file for upload. Here, the file is encrypted to ensure security: an RSA-based encryption approach is used, often in combination with RSA for efficiency. In this hybrid scheme, the file content is encrypted using a randomly generated RSA session key, and then this RSA key is encrypted using the Server's or per-file RSA public key.



5. JSP Dashboard

Displays the uploaded encrypted data along with file details and timestamps. It also shows the blockchain transaction status, verification results, pending registration details, and overall project modules. The dashboard acts as the user interface for monitoring stored data, status, and user details etc.

6. MySQL Database

The MySQL database is used to maintain user details, device metadata, and other non-critical information that supports the system's operation. While data is securely stored on the blockchain to guarantee immutability and tamper-resistance, the database handles supporting records such as login credentials, user roles, and system logs. This hybrid storage approach ensures both efficient system performance and strong data security by separating critical data from auxiliary information.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results show that 182 out of the top 500 websites (36.4%) use cloud storage services, and all 28 analyzed websites with direct file upload functionality contained at least one security vulnerability. A total of 79 previously unknown vulnerabilities were identified and responsibly disclosed to the affected platforms. The analysis highlights common security weaknesses in cloud-based file upload mechanisms and proposes effective mitigation strategies. Overall, the study demonstrates the importance of strengthening security practices for direct cloud file uploads in modern web applications.



Fig. 2: User Login

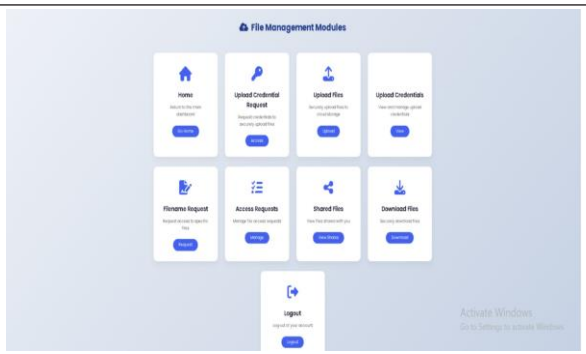


Fig. 3: User Dashboard

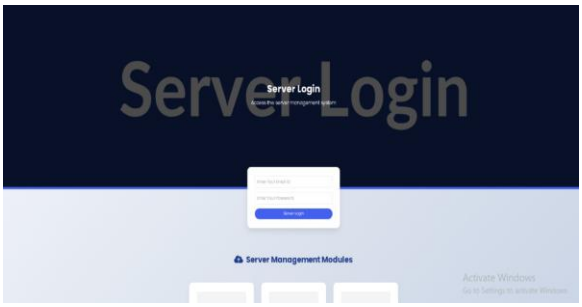


Fig. 4: Server Login



Fig. 5: Table Data

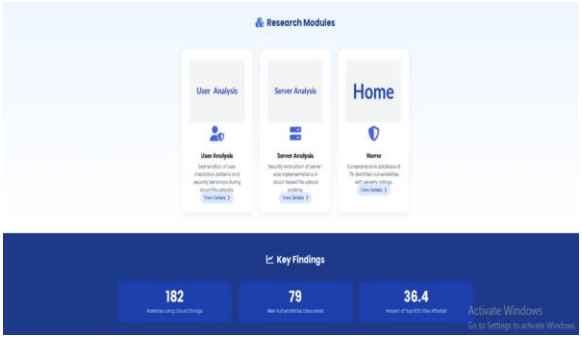


Fig. 6: Research Module

V. CONCLUSION

In conclusion, this project provides a secure and efficient framework for directly uploading files from web users to cloud storage while addressing critical security vulnerabilities. By implementing RSA-based encryption and a robust credential management system, it ensures data confidentiality, integrity, and authorized access throughout the file lifecycle. The User and Server modules work in coordination to manage authentication, upload credential issuance, decryption key distribution, and callback verification, effectively mitigating vulnerabilities such as credential misuse, file stealing, file overwriting, and callback spoofing. The systematic study of the three-stage upload process allows the identification and prevention of security risks that traditional methods may overlook. The architecture is designed to be scalable, reliable, and user-friendly, supporting secure file uploads, downloads, and key management. Overall, this project demonstrates a comprehensive approach to securing cloud storage interactions, providing a strong foundation for future enhancements and safe cloud-based data management for modern web applications.

REFERENCES

[1] P. Yang, N. Xiong, and J. Ren, "Data security and privacy protection for cloud storage: A survey," IEEE Access, vol. 8, pp. 131723–131740,2020.

[2] (2021). Gmail Statistics, Users, Growth and Facts for 2021. [Online].Available: https://saasscout.com/statistics/gmailstatistics/

[3] (2019). Reddit's 2019 Year in Review. [Online]. Available: https://www.redditinc.com/blog/reddits-2019-year-in-review/

[4] R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie,"Cloud storage reliability for big data applications: A state of the art survey," J. Netw. Comput. Appl., vol. 97, pp. 35–47,Nov.2017.

[5] T. Lee, S. Wi, S. Lee, and S. Son, "FUSE: Finding file upload bugs via penetration testing," in Proc. Netw. Distrib. Syst. Secur. Symp., 2020,pp. 1–11.

[6] Y. Chen, Y. Li, Z. Pan, Y. Lu, J. Chen, and S. Ji, "URadar: Discovering unrestricted file upload vulnerabilities via adaptive dynamic testing," IEEE Trans. Inf. Forensics Security, vol. 19, pp. 1251–1266,2024.

[7] C. Zuo, Z. Lin, and Y. Zhang, "Why does your data leak? Uncovering the data leakage in cloud from mobile apps," in

Proc. IEEE Symp. Secur. Privacy (SP), May 2019, pp. 1296–1310.

[8] Y. Zhou, L. Wu, Z. Wang, and X. Jiang, "Harvesting developer credentials in Android apps," in Proc. 8th ACM Conf. Secur. Privacy Wireless Mobile Netw., Jun. 2015, pp. 1–12.

AUTHORS Profile



Ms. S. SABIYA SULTHANA has received her B. Tech in CSE from JNTUA in 2014, M. Tech in Computer Science and Engineering from JNTUA in 2017 and She has been dedicated to the teaching field for the last 2 years. At present she is working as Assistant Professor in CSE Department at Guru Nanak Institutions Technical Campus, Ibrahimpatnam, Telangana, India.



Ms. BANDI SANJUNA pursuing B. Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.



Ms. VINJAM VANISRI pursuing B. Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.



Mr. BANDA NITHIN REDDY pursuing B. Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.



Mr. BONAGIRI NIKHIL pursuing B. Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.