A Security Framework for Direct User-to-Cloud Upload Systems with Automation for Vulnerability Detection

S.Srimathi[1], K.Shanthini Smilin [2], G.Gowri[3]

Assistant professor, Surya engineering college, erode.

Assistant professor, Surya engineering college, erode.

Teaching assistant, Surya engineering college, erode.

Srimathi257@gmail.com, smilinshanthini@gmail.com, gandhigowri65@gmail.com

ABSTRACT

Directly uploading files from user-to-cloud ,it becomes a fundamental component of the modern web application. It is based on their efficiency, scalability, reduced server load. By passing traditional web server, this model usually upload files directly to cloud. It also improves performance and introduces new security risks involving temporary credentials, key binding and callback verification. This study presents a structured analysis of these risks by proposing a six-category vulnerability framework based on real-world cloud upload workflows. The large scale evaluation of high traffic website reveals the critical misconfigurations including unlimited credential issuances, long lived upload tokens, weak file validation, unauthorized file access, and callback spoofing. The base paper comprehensively mapped these vulnerability through manual testing, the absence of an automated detection mechanism it remains a significant research gap. Therefore, the future work proposed in this study is the development of an Automated Security Testing Tool for direct cloud uploads. It aims to enable scalable, continuous and systematic detection of misconfigurations and vulnerabilities within cloud integrated web architectures.

Index terms:

Cloud Storage, Direct Upload, Temporary Credentials, Web Security, IAM Misconfiguration, Callback Spoofing, Automated Security Testing

---

1. INTRODUCTION

The ever increasing popularity of websites, storing and managing user generated data become the significant challenge. In the existing system, under the cloud storage service the new file uploading scenario has been introduces, where the files are directly uploaded from web users to the cloud. There are three critical stages in this new file upload scenario i) uploading credential requesting and dispatching, ii) file uploading and verification, and iii) callback notification and response. First the process involves the authentication interactions among these three roles and the each role should responsible for its own security authentication, if it fails to provide a proper identification for the authentication for a user, it opens up the possibilities for attackers to forget the users identity. The second process is new authorization mechanism. Finally the new scenario involves information synchronization among the three roles. In existing it also perform the study of the security risks and directly uploading files web users to a cloud storage service. The six types of the vulnerability also introduced: unrestricted upload credentials acquisition, upload credentials validity flaw, unrestricted file types and file size, file overwriting, file stealing, and callback notification spoofing.

The application server generates short live credentials that allows the user to upload files directly to the cloud, and it eliminates the server which is overhead. This architectural model introduces unique security challenges. The interaction is commonly between client, server and cloud provider. The proposed system of the existing one is introducing the automation as the primary direction for future research.

2.RELATED WORK

The existing research on cloud security it examines the storage bucket misconfigurations, IAM privilege escalation, and API key leaks. The web security it always focus on server side file upload vulnerabilities such as MIME spoofing or arbitrary upload flaws. No prior research specifically evaluates temporary upload credentials (presigned URLs, upload tokens). The call back manipulation in the cloud uploads and it remains largely unexplored. Here, there is no automated vulnerability scanners currently exists for direct-to-cloud upload workflows. In this research it addresses the significant gap as cloud specific upload security literature. The research gap for the existing system is lack of automated security testing tools. Here, it is updated as automatic security for the cloud storage rather then the manual way.

The base paper provides the strong insights through the manual testing ,its primary limitation is the lack of automated tools for continuous and scalable security analysis. The proposed system it uses some security ways to protect the cloud storage system using security tool.

3.EXISTING SYSTEM

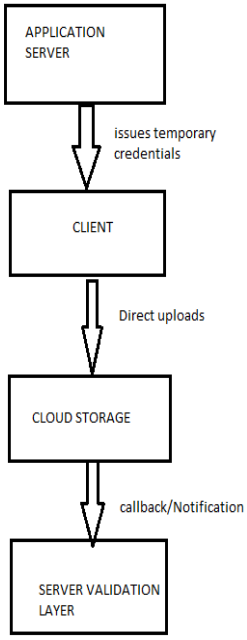
The first comprehensive security study on websites, it uses the direct user to cloud file upload system. The modern web application frequently integrate cloud storage platforms and it also used to handle large-scale file uploads. In this architecture, it analyze how user files move from client device to the cloud storage and it also identifies the communication path, security controls and potential bottlenecks. The main concept of the existing system is creating a new vulnerability classification frameworks and it identifies the major risk in cloud upload systems.

3.1 VULNERABILITY ANALYSIS

It provides an in-depth examination of the security vulnerability discovered in modern direct user to cloud upload workflows. It is structured into six categories theses are based on the phases of credential issuance, file upload, and callback handline. Each category reflects weaknesses identified in real-world implementations and rooted in cloud provider misconfigurations, improper API usage, or insecure application logic. The first as , Direct upload system require temporary credentials like presigned URLs, upload tokens, short-lived IAM keys. The credential issued only after strong authentication and strict rate limiting. The second as, long lived upload credentials -Temporary credentials are meant to be short lived and single use. Third as, Insufficient file validation it depends solely on client side checks for the MIME type verification, file extension checks, file size limits. Fourth, as file overwriting - In the cloud the objects are stored using keys. The key name is controlled entirely by the client without mandatory server validation. If the server does not securely bind a file path to a user identity then the attacker can easily overwrite. Fifth, as unauthorized file reading/listing then the sixth, as call back spoofing, after the successful uploading the file, the client or cloud provider send a callback to the server it confirms upload success, register metadata, trigger backend processing.

3.2 DIRECT UPLOAD SYSTEM ARCHITECTURE

In this mechanism it follow the three major phases: i)credential issuance ii)direct file uploads iii)callback/notification. The first one contains the presigned URLs, policy documents, short-lived access keys. The second contains the uploaded file directly to the cloud storage endpoint. The third contains the cloud provider or client sends uploads results back to the server.



3.3 LARGE SCALE REAL WORLD SECURITY EVALUATION

In the existing system it selected the 500 top ranked websites. In the 500 websites 182 websites used cloud storage that is detected through DNS analysis, developer tools and network traces. Then it focused on 28 websites that supports direct user file uploads using presigned URLs, upload tokens, or temporary credentials. It use ethical evaluation methodology.

3.4 DISCOVERY OF 79 REAL VULNERABILITIES

The discovery of 79 real world vulnerabilities across website that implements direct user to cloud upload mechanisms. It is used to identifies a combination of static policy inspection, dynamic adversarial testing, and controller attacker-victim simulations. There are 79 total security flaws. All the 28 evaluated websites are vulnerable and it includes the severe cases as file overwriting, full-bucket read access, call back, forgery ,unlimited credential framing.

3.5 ROOT CAUSE ANALYSIS

In the 79 real-world vulnerabilities across 28 websites demonstrate that the security weakness in direct user to cloud upload systems. The following root cause analysis categorize the underlying failure into technical, architectural and human factors. the vulnerabilities are: overly permissive cloud permissions, weak client-side validation, misunderstanding of cloud IAM policies, insecure callback implementation.

3.6 MITIGATION AND BEST PRACTICES

The discovery of multiple vulnerabilities in direct user-to-cloud upload systems highlights the need for strong security measures when implementing temporary credentials, object storage policies, and callback workflows. This section proposes a set of systematic mitigation strategies and best practices that developers and cloud administrators should adopt to prevent misuse, minimize attack surfaces, and enforce secure-by-default behavior. These recommendations are derived from the root cause analysis and the 79 real-world vulnerabilities identified in the study.

Then the suggestion for the solution is rate limiting credential issuance, shorter credential validity, stricter IAM permissions, cryptographically signed callback, server side file validation.

4.PROPOSED SYSTEM

In the base paper it provides the strong insights through manual testing, here the primary limitations is the lack of automated tools for continuous and scalable security analysis. The main concept of the future work is development of an Automated Cloud Upload Security Testing Tool.(ACUST).

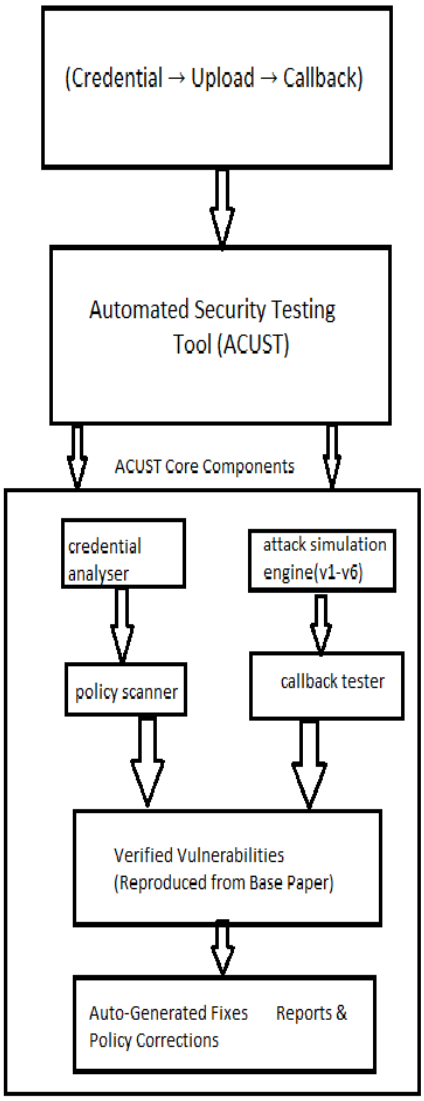
The base paper conducted a large-scale, manual evaluation of direct user-to-cloud upload systems and identified 79 real-world vulnerabilities across 28 websites. Although the manual approach successfully revealed critical security issues, it remains limited by human effort, time, and lack of scalability. To address this limitation, the proposed Automated Security Testing Tool (ACUST) is applied to the base paper's methodology to improve reproducibility, accuracy, and scalability. The purpose of applying ACUST : i) it enables the reproduction of the 79 vulnerabilities. ii) it automatically re-tests each vulnerability category, iii) it test hundred or thousands of websites at once. iv) continuous monitoring and website can be periodically scanned for regressions or new misconfigurations.

The automated tool evaluates the three stages of the base paper: i) Credential Generation ii) Direct File Upload iii) Callback Handling. After the three stages it automatically evaluate the six vulnerability classes. It automatically evaluates the six vulnerability stages.

Vulnerability ID	Automated Test Performed
V1	Token farming, rate-limit bypass
V2	Credential reuse, expiry manipulation
V3	Uploading invalid file types/sizes
V4	Path manipulation & overwrite attempts
V5	Unauthorized GET/LIST access
V6	Forged callback simulation

ATOMATED CLOUD UPLOAD SECURITY TESTING TOOL.(ACUST)

In the proposed system, it identify the defects unsafe upload credentials , analyzes upload policies, performs dynamic security testing, monitors cloud storage misconfigurations, generates automated security reports.



ALGORITHM

Multi-Stage Cloud Upload Vulnerability Detection (MCUVD)

Input:

Website URL with direct cloud upload functionality

Output:

List of detected vulnerabilities with severity and mitigation

Step-by-Step Algorithm

Stage 1: Credential Extraction

Stage 2: Automated Attack Simulation

Stage 3: Risk Classification & Reporting

Algorithm Fits the Base Paper, Because The base paper's vulnerabilities fall into six core categories.

The MCUVD algorithm tests each category automatically using structured attack simulation.

5.CONCLUSION

Nowadays, more and more websites uses cloud service for file storage. Directly uploading files from web users to cloud storage is one of the most mainstream scenarios. In this paper, we systematically analyze the security risks . We conduct an in-depth analysis of the three critical stages in this scenario, including upload credential requesting and dispatching, file uploading and verification,

and callback notification and response.

Direct user-to-cloud file upload systems offer significant performance benefits but introduce complex security challenges. The base paper identified widespread vulnerabilities across real-world platforms and proposed a structured security framework.

This journal paper builds on that foundation by identifying a key research gap—the absence of automated security testing—and proposing a comprehensive automated tool as future work. Developing such a tool will enable scalable and continuous evaluation of cloud-upload workflows, significantly advancing the security of modern cloud-integrated 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/gmail-statistics/

[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. 5–47, 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.

[9] M. Meli, M. R. McNiece, and B. Reaves, "How bad can it git? Characterizing secret leakage in public GitHub repositories," in Proc.Netw. Distrib. Syst. Secur. Symp., 2019, pp. 1–18.

[10] H. Wen, J. Li, Y. Zhang, and D. Gu, "An empirical study of SDK credential misuse in iOS apps," in Proc. 25th Asia–Pacific Softw. Eng.Conf. (APSEC), Dec. 2018, pp. 258–267.

[11] X. Wang, Y. Sun, S. Nanda, and X. F. Wang, "Credit karma: Understand ing security implications of exposed cloud services through automated capability inference," in Proc. 32nd USENIX Secur. Symp. (USENIX Secur.), 2023, pp. 6007–6024.

[12] T. Bhatia and A. K. Verma, "Data security in mobile cloud computing paradigm: A survey, taxonomy and open research issues," J. Supercom put., vol. 73, no. 6, pp. 2558–2631, Jun. 2017.

[13] Z. Xia, N. N. Xiong, A. V. Vasilakos, and X. Sun, "EPCBIR: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing," Inf. Sci., vol. 387, pp. 195–204, May 2017.

[14] D. He, N. Kumar, S. Zeadally, and H. Wang, "Certificateless provable data possession scheme for cloud-based smart grid data management systems," IEEE Trans. Ind. Informat., vol. 14, no. 3, pp. 1232–1241, Mar. 2018.

[15] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," IEEE Trans. Inf. Forensics Security, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

[16] J. Cable, D. Gregory, L. Izhikevich, and Z. Durumeric, "Stratosphere: Finding vulnerable cloud storage buckets," in Proc. 24th Int. Symp. Res.Attacks, Intrusions Defenses, Oct. 2021, pp. 399–411.

[17] Academia.edu. Accessed: 2022. [Online]. Available: https://www.academia.edu/

[18] Uploading to Amazon S3 Directly From a Web or MobileApplication.Accessed:2022.

[Online].

Available:https://aws.amazon.com/blogs/compute/uploading-to-amazon-s3 directly-from-a-web-or-mobile-application/

[19] Uploading Objects to OSS Directly From Clients. Accessed: 2022. [Online]. Available: https://help.aliyun.com/zh/oss/use-cases/uploading objects-to-oss-directly-from-clients/

[20] Comparing Aws Account Root User Credentials and IAM User Creden tials. Accessed: 2022. [Online]. Available: https://docs.aws.amazon.com/ accounts/latest/reference/root-user-vs-iam.html

[21] Introduction to Resource Access Management—Alibaba Cloud Document Center. Accessed: 2022. [Online]. Available: https://

www.alibabacloud.com/help/en/resource-access-management

[22] Use STS Temporary Access Credentials to Access OSS. Accessed: 2022. [Online]. Available: https://help.aliyun.com/document detail/ 100624.html

[23] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," IEEE Trans. Dependable Secure Comput., vol. 9, no. 6, pp. 903–916, Nov. 2012.

[24] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "FADE: Secure overlay cloud storage with file assured deletion," in Proc. Int. Conf. Secur. Privacy Commun. Syst. Cham, Switzerland: Springer, Jan. 2010, pp. 380–397.

[25] M. Chatterjee, P. Datta, F. Abri, A. Siami Namin, and K. S. Jones, "Abuse of the cloud as an attack platform," in Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC), Jul. 2020, pp. 1091–1092.