



ENHANCING SECURITY IN DIRECT FILE UPLOAD TO CLOUD STORAGE SERVICES

Mashaeil Shaik¹, Dr. Lalitha Saroja CH², Ruqiya Fatima³

¹PG Scholar, Department of AI&DS, Shadan Women's College of Engineering and Technology, Hyderabad, mashaeilshaik44@gmail.com

²Assoc Professor, Department of CSE, Shadan Women's College of Engineering and Technology, lalithasarojathota@gmail.com

³Asst Professor, Department of CSE, Shadan Women's College of Engineering and Technology, ruqiyafatima2716@gmail.com

ABSTRACT:

Websites have started to allow users to upload files directly to cloud platforms due to the growing reliance on cloud storage services for managing massive amounts of user data. Due to the participation of several parties, including online users, web servers, and cloud storage providers, this strategy presents additional security issues even if it offers more ease and scalability. We give the first thorough security analysis of this direct upload approach in this work. We do extensive testing across the top 500 Alexa-ranked websites after conducting a thorough research that identifies six different kinds of vulnerabilities. According to our research, 182 websites (36.4%) make use of cloud storage services, and a detailed examination of 28 well-known websites with upload capabilities reveals that every one of them has at least one vulnerability. We found 79 previously undiscovered vulnerabilities in total, which we appropriately notified to the relevant platforms, including well-known sites like Google, Reddit, and CSDN. The favorable reactions demonstrate the usefulness of our conclusions. We investigate these problems' underlying causes in further detail and offer practical solutions. The goal of this effort is to help academics and developers create better secure online applications by offering insightful information about the security implications of cloud-based file uploads.

KEYWORDS: —Cloud storage service, web security, upload credential, vulnerability detection.

1. INTRODUCTION

As a result, handling and storing large amounts of data created by users becomes extremely difficult. For instance, hundreds of millions of people visit well-known websites like Google and Reddit, according to statistical data. Among Google services, Gmail alone has more than 1.8 billion users. There are over 430 million active users on Reddit each month. As a result, it is critical to solve the data storage and management concerns of the aforementioned large users. Because of its flexibility, scalability, and pay-as-you-go storage options, cloud storage services have become the go-to option for managing and storing website data. These features meet the requirements of websites in a range of sizes and sectors. This new file upload scenario typically involves three crucial steps: (1) obtaining and dispatching upload credentials; (2) uploading and verifying files; and (3) callback notice and response. In stage (1), the client sends an HTTP request to the web server to request an upload credential when a user wishes to upload a file to a website. After verifying the user's identity, the web server should provide the upload credentials if it is successful. In step (2), the client uploads the file straight to the cloud storage provider using the credentials. To let the web server know the outcome of the file upload, either the client or the cloud storage service should send a callback message in stage (3). This novel scenario removes the requirement for the server to handle data storage or the transit of submitted files, in contrast to conventional techniques where the web server saves user files locally or acts as a middleman to send files to cloud services. For web servers, this lowers transmission overhead and provides ease. But it also creates the

following new attack surfaces. For example, if a user's identity is not properly authenticated by the web server or cloud storage service, attackers may be able to fake the user's identity. Second, a new permission mechanism—upload credentials—is introduced by the new scenario. Privilege escalation vulnerabilities may result from improperly configured upload credentials or policies. Adversaries could leverage these flaws, for example, to steal or alter other users' files in cloud storage systems. Lastly, information synchronization across the three roles is a feature of the new scenario. Attackers can then deceive the web server by forging or manipulating callback notifications. Prior research on cloud storage security has concentrated on mobile applications that have problems with data management and user credentials abuse. To the best of our knowledge, however, no research has been done on the security concerns of this novel situation. Analyzing this scenario's security threats presents two major issues. 1) It is difficult to comprehend the intricate mechanisms of interaction between various jobs and identify any weaknesses. A typical online user cannot see many of these exchanges. 2) Determining whether an interaction mechanism's actual deployments or implementations are insecure is a difficult task. The diversity of cloud service providers and websites, each of which often has unique implementations or deployments, contributes to the complexity. We provide the first comprehensive analysis of the security concerns associated with online users directly uploading data to cloud storage services in this research. Based on the public development guides offered by cloud service providers, we do a thorough study of the three crucial



steps in this situation in order to handle the aforementioned issues. These flaws, which include resource-consumption assaults, privacy leakage, sensitive data overwriting, etc., have the potential to seriously impact online users, web servers, and cloud storage services. Keep in mind that, in contrast to user credentials, upload credentials are usually created momentarily based on user credentials in accordance with predetermined security regulations. To put it another way, the upload credentials that many web users acquire are usually produced from the same cloud service account. Users are given the same permissions from the standpoint of the cloud storage service. Limitations on permitted file kinds, sizes, storage locations, credential expiration periods, etc. may be included in the policies. The web server verifies the user's upload rights and login status. The web server will provide the upload credentials to the user in two usual methods if the verifications are successful. Keep in mind that, in contrast to user credentials, upload credentials are usually created momentarily based on user credentials in accordance with predetermined security regulations. To put it another way, the upload credentials that many web users acquire are usually produced from the same cloud service account. Users are given the same permissions from the standpoint of the cloud storage service. Limitations on permitted file kinds, sizes, storage locations, credential expiration periods, etc. may be included in the policies. The web server verifies the user's upload rights and login status. The web server will provide the upload credentials to the user in two usual methods if the verifications are successful. The cloud storage service should perform security checks on the uploaded file and the upload credential. Security checks for the uploaded file could include: 1) if it can replace an existing file; and 2) the file's characteristics, such as whether its size or type complies with upload policy restrictions. The cloud storage provider should verify the authenticity of the upload credential. For example, the upload request should not be permitted if the user's upload credentials have expired.

2. STIMULATING

In the three crucial phases—upload credential management, file uploading and verification, and callback notification handling—it seeks to detect and reduce any security threats. The project tackles vulnerabilities including data manipulation, file overwriting, illegal access, and credential abuse. It guarantees safe user, web server, and cloud service synchronization, authorization, and authentication. In the end, the project improves large-scale cloud-based web applications' data integrity, confidentiality, and dependability.

3.OBJECTIVE

In order to find any security flaws, it focuses on examining the three crucial phases of the upload

process: credential generation, file verification, and callback response. Through the implementation of robust authentication and policy enforcement methods, the project seeks to avoid risks including unauthorized access, credential abuse, file overwriting, and data theft. Through safe communication and synchronization, it also aims to increase confidence between users, web servers, and cloud platforms. Building a strong foundation that enhances data security, integrity, and confidentiality in contemporary cloud-based online environments is the ultimate objective.

3.1 What The Project Is About

However, this presents serious security issues that have not yet been thoroughly investigated. Because users, web servers, and cloud storage interact in several stages, there may be authentication holes that let hackers pose as actual users. Upload credentials that are incorrectly scoped or misconfigured increase the risk of privilege escalation and illegal access. Sensitive data theft, file overwriting, and storage misuse can arise from lax regulation of file types, sizes, and ownership. Data integrity is at risk due to callback notifications from the cloud to the server being spoofable. There are gaps in the security of direct-to-cloud uploads since existing research mostly concentrates on data management or mobile apps. Without strong encryption and key management, user data confidentiality and integrity are also at risk. To find vulnerabilities, implement secure credential management, and guarantee secure file uploads and downloads, a methodical methodology is needed. The system can protect data confidentiality and stop unwanted access by using RSA-based encryption and controlled decryption. Authentication, credential issuance, file encryption, decryption key distribution, and callback verification must all be securely managed by modules. Protecting user data while facilitating effective direct-to-cloud uploads is the aim.

4. EXISTING SYSTEM:

- Instead of using conventional server-side storage, users upload files straight to the cloud from the client side in this configuration. Due to the inclusion of several roles—online users, web servers, and cloud service providers—this architecture presents new security issues despite being effective and scalable. Credential issuance, file upload and verification, and callback notification are the three primary phases of the upload procedure.
- A new scenario in the present architecture eliminates the necessity for the website server to function as a middleman by enabling users to upload files straight from their browsers to cloud storage. Due to the participation of several parties—online users, web servers, and cloud storage providers—this approach improves efficiency and lowers server strain, but it also



creates new security threats. Nevertheless, the current system's synchronization, authorization, and authentication processes are insufficiently secure.

4.1 Existing System Disadvantages:

- There are issues with the present edge computing architecture.
- Attackers have the ability to replace existing data by uploading a file with the same name.
- From user upload to cloud confirmation, the system does not guarantee data authenticity and integrity.

4.2 THE PROPOSED SYSTEM

This direct interface adds many roles—users, web servers, and cloud services—all of which require safe coordination, in contrast to typical techniques that route data through web servers. Improper handling of upload credentials, unlimited file kinds and sizes, file overwriting risks, unauthorized file access, and fake callback alerts are among the six main categories of vulnerabilities identified by the study. 79 new vulnerabilities were found in all. This study is important because it highlights the new risks associated with direct-to-cloud uploads and provides advice on how to avoid these problems by enhancing access control, authentication, and secure communication methods.

4.2.1 Proposed System Advantage

- uses access control and robust encryption to stop unwanted file downloads and access.
- It facilitates low-overhead, scalable cloud-based deployment.

5. METHODOLOGIES

1. Design of User Interfaces

The user can log in to the server directly if they have previously left; otherwise, they must register their information, including their username, password, email address, city, and country. To manage upload and download rates, the database will generate an account for every user. The user ID will be used as the name. To access a particular website, you often need to log in. The query will be searched and displayed.

2. User

To ensure that the data is private and unchangeable, the User module encrypts the file content using RSA-based encryption (or hybrid RSA+AES for efficiency) after obtaining the credential. In order to safely access encrypted data, the module also handles download requests, which either get files from the cloud or ask the server for the matching decryption key. It also does controlled, non-destructive systematic checks for the six vulnerability classes (V1–V6), including handling callback notifications, preventing unauthorized access, and confirming file type constraints. **3. Server** It creates RSA key pairs or encrypts AES keys for every file, checks user IDs, and provides scoped and time-limited upload privileges. In order to guard

against spoofing or manipulation, the server additionally verifies cloud callbacks by examining signatures, nonces, and file metadata correctness. Additionally, it enforces file ownership and integrity regulations, logs all activity for audits, and mediates the safe distribution of decryption keys to only authorized users. In order to offer secure file storage, preserve confidentiality and integrity, and methodically eliminate all six vulnerability categories in the direct-to-cloud upload procedure, the User and Server modules collaborate.

4. File Vulnerabilities & RSA

When a user module asks the server for an upload credential, the server authenticates the user and provides a scoped, time-limited credential. In order to determine if the server improperly issues or accepts controlled requests with invalid, expired, or role-switched credentials, the method first checks for unrestricted credential acquisition (V1) and credential validity defects (V2). The User module gets the file ready for upload after obtaining a valid credential. In order to guarantee security, the file is encrypted using an RSA-based encryption technique, frequently in conjunction with RSA for effectiveness. A randomly generated RSA session key is used in this hybrid approach to encrypt the file content. The server's or per-file RSA public key is then used to encrypt the RSA key.

6. TECHNIQUE ALGORITHM USED

6.1 Proposed Algorithm

➤ Systematic study of the Security risks, RSA

In addition to checking for six significant vulnerabilities (V1–V6) and guaranteeing data confidentiality with RSA encryption, the project's suggested methodology is made to methodically safeguard the direct-to-cloud file upload procedure. The User module first requests an upload credential from the Server, which verifies the user's identity and provides a time-limited, scoped credential. In order to determine if the server improperly issues or accepts controlled requests with invalid, expired, or role-switched credentials, the method first checks for unrestricted credential acquisition (V1) and credential validity defects (V2). The User module gets the file ready for upload after obtaining a valid credential. In order to guarantee security, the file is encrypted using an RSA-based encryption technique, frequently in conjunction with RSA for effectiveness. A randomly generated RSA session key is used in this hybrid approach to encrypt the file content. This RSA key is then encrypted using the server's or per-file RSA public key. This guarantees that the original file can only be accessed by the authorized server or user who has the matching RSA private key. The technique attempts safe test uploads using boundary cases, colliding filenames, or modified object keys to check for unlimited file kinds and sizes (V3), file overwriting (V4), and file theft (V5) during the upload stage. In order to ensure that the Server verifies



signatures, nonces, and coherence with stored information, the algorithm lastly watches and simulates messages from the cloud to the Server during the callback stage. It then checks for callback spoofing (V6) by delivering constructed or tampered notifications. In order to reduce false positives, the system logs evidence for every vulnerability check, securely verifies results, and adaptively improves testing depending on observed behaviors. Through safe key management and RSA-based encryption, our methodology guarantees that all six vulnerabilities are methodically examined while maintaining the confidentiality and integrity of uploaded data.

6.2 EXISTING SYSTEM

➤ Cloud Strobe

The term "cloud storage" refers to an online data management system that has grown more and more important as website traffic and user-generated content have increased dramatically. Cloud storage services offer a scalable, adaptable, and affordable substitute for conventional local storage systems as websites struggle to manage enormous volumes of user data. A new scenario in the present architecture eliminates the necessity for the website server to function as a middleman by enabling users to upload files straight from their browsers to cloud storage. Due to the participation of several parties—online users, web servers, and cloud storage providers—this approach improves efficiency and lowers server strain, but it also creates new security threats. In order to provide high availability, dependability, scalability, and fault tolerance, the algorithm decides how data is distributed, assigned, duplicated, and accessed inside the cloud architecture. In order to prevent data loss in the event of server failures, the cloud storage algorithm may divide a file uploaded by a user into smaller blocks, distribute these blocks over several storage nodes, and produce replicas or parity information. The program finds the appropriate blocks during data retrieval, reconstructs the original file if needed, and then sends it to the user. Such algorithms are used by well-known cloud services to handle enormous amounts of data while offering quick access and constant availability. Cloud storage methods improve fault tolerance, minimize storage overhead, optimize block placement, and guarantee effective data recovery in edge and cloud contexts in distributed storage systems like EdgeHydra.

7. SYSTEM ARCHITECTURE

The User Module and the Server Module, which are coupled to the Cloud Storage Service, are the two fundamental components of the architecture. Users may register, log in, obtain upload credentials, encrypt files using the RSA technique, and safely upload them to the cloud using the User Module, which serves as the client interface. Additionally, after receiving server permission, users can download encrypted files and request decryption keys. The central control unit

called the Server Module is in charge of creating upload credentials, keeping decryption keys, confirming user identities, and confirming authorization for every request. Encrypted files are stored, upload credentials are verified, and callback alerts on upload status are sent to the server using the Cloud Storage component.

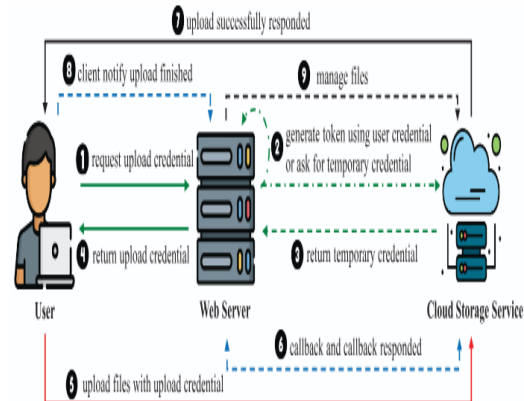


FIG:7 System Architecture

To avoid interception or manipulation, encrypted HTTPS protocols are used for all module-to-module communication. Additionally, the system uses stringent authentication, encryption, and access control measures to mitigate six significant vulnerabilities: credential abuse, file overwriting, file theft, uncontrolled upload access, credential validity issues, and callback spoofing. The architecture provides a solid basis for safe cloud-based storage operations by combining tiered permission with cryptographic security to guarantee data confidentiality, integrity, and reliability across the whole file lifecycle.

8. Results



Fig 8.1 Index page

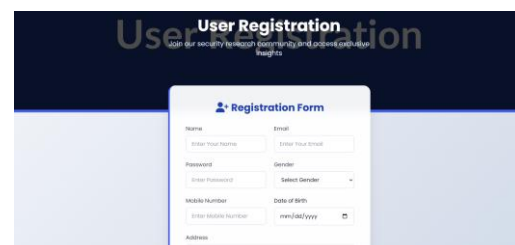
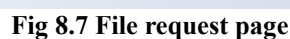


Fig 8.2 Registration page



More sophisticated security, usability, and scalability features might be added to the project in the future. One improvement may be to incorporate multi-factor authentication for users, which would provide an additional degree of protection prior to providing upload credentials or decryption keys. Implementing role-based access control, which gives certain user types specific capabilities to upload, download, or manage data, might be another enhancement. In order to spot suspect activity, including illegal file access or credential misuse, the system can additionally offer real-time monitoring and anomaly detection. Large-scale file uploads can be encrypted more effectively by combining sophisticated encryption techniques like ECC (Elliptic Curve Cryptography) with RSA. Future iterations could provide cross-platform file access, allowing users on desktop and mobile devices to safely upload and download data. Administrators may monitor all activities and swiftly identify security breaches by using automatic logging and auditing. Attacks like file theft and callback spoofing might be proactively stopped by integrating AI-based threat detection. File versioning is another feature that the system may use to avoid inadvertently erasing critical data. Users may have complete control over the security of their files using end-to-end encryption and user-managed keys. File management may be made easier and the user experience enhanced by adding user-friendly dashboards. For increased efficiency, bulk uploads and downloads are supported by the system. API-based interaction with third-party apps for smooth cloud services might be a future development. Lastly, load balancing and performance optimization may guarantee that the system stays dependable and quick even with millions of users. With these improvements, the system will be more

683



user-friendly, scalable, and secure, meeting the needs of contemporary cloud storage.

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.
- [9] M. Meli, M. R. McNiece, and B. Reaves, "How bad can it get? 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: Understanding 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. Supercomput.*, 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 Mobile Application. 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/uploadingobjects-to-oss-directly-from-clients/>
- [20] Comparing Aws Account Root User Credentials and IAM User Credentials. 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.
- [26] (2023). Amazon S3 Simple Storage Service Pricing—Amazon Web Services. [Online]. Available: https://aws.amazon.com/s3/pricing/?nc1=h_ls
- [27] (2023). Azure Blob Storage Pricing — Microsoft Azure. [Online]. Available: <https://azure.microsoft.com/enus/pricing/details/storage/blobs/>
- [28] (2023). OSS (Object Storage Service) Pricing & Purchasing Methods—Alibaba Cloud. [Online]. Available: <https://www.alibabacloud.com/product/oss/pricing>



- [29] M. Wachs, L. Xu, A. Kanevsky, and G. R. Ganger, "Exertion-based billing for cloud storage access," in Proc. 3rd USENIX Workshop Hot Topics Cloud Comput. (HotCloud 11), Jun. 2011, p. 7.
- [30] G. Hong, M. Wu, P. Chen, X. Liao, G. Ye, and M. Yang, "Understanding and detecting abused image hosting modules as malicious services," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Nov. 2023, pp. 3213–3227.
- [31] Y. Lv, W. Shi, W. Zhang, H. Lu, and Z. Tian, "Don't trust the clouds easily: The insecurity of content security policy based on object storage," IEEE Internet Things J., vol. 10, no. 12, pp. 10462–10470, Jun. 2023.
- [32] L. Weichselbaum, M. Spagnuolo, S. Lekies, and A. Janc, "CSP is dead, long live CSP! On the insecurity of whitelists and the future of content security policy," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2016, pp. 1376–1387.
- [33] S. Calzavara, A. Rabitti, and M. Bugliesi, "Content security problems: Evaluating the effectiveness of content security policy in the wild," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2016, pp. 1365–1375.
- [34] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Crosssite scripting (XSS) attacks and mitigation: A survey," Comput. Netw., vol. 166, Jan. 2020, Art. no. 106960.
- [35] G. Pellegrino, M. Johns, S. Koch, M. Backes, and C. Rossow, "Deemon: Detecting CSRF with dynamic analysis and property graphs," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2017, pp. 1757–1771.
- [36] D. Kim and H. Kim, "Performing clickjacking attacks in the wild: 99% are still vulnerable!," in Proc. 1st Int. Conf. Softw. Secur. Assurance (ICSSA), Jul. 2015, pp. 25–29.
- [37] D. Bates, A. Barth, and C. Jackson, "Regular expressions considered harmful in client-side XSS filters," in Proc. 19th Int. Conf. World Wide Web, Apr. 2010, pp. 91–100.
- [38] M. Mohammadi, B. Chu, H. R. Lipford, and E. Murphy-Hill, "Automatic Web security unit testing: XSS vulnerability detection," in Proc. IEEE/ACM 11th Int. Workshop Autom. Softw. Test (AST), May 2016, pp. 78–84.
- [39] Amazon. (2022). Cloud Object Storage—amazon S3. [Online]. Available: <https://aws.amazon.com/s3/>
- [40] Cloud Computing Services — Microsoft Azure. Accessed: 2022. [Online]. Available: <https://azure.microsoft.com/>
- [41] Object Storage Data Processing-Cos Data Processing-Data Processing Solution-Tencent Cloud. Accessed: 2022. [Online]. Available: <https://cloud.tencent.com/product/cos>
- [42] Object Storage BOS. Accessed: 2022. [Online]. Available: <https://cloud.baidu.com/product/bos.html>
- [43] About Object Operations. Accessed: 2022. [Online]. Available: <https://help.aliyun.com/document-detail/31977.html>
- [44] Amazon S3 Rest API Introduction—Amazon Simple Storage Service. Accessed: 2022. [Online]. Available: <https://docs.aws.amazon.com/zh-cn/AmazonS3/latest/API/Welcome.html>
- [45] J. Qian, S. Hinrichs, and K. Nahrstedt, "ACLA: A framework for access control list (ACL) analysis and optimization," in Communications and Multimedia Security Issues of the New Century: IFIP TC6 / TC11 Fifth Joint Working Conference on Communications and Multimedia Security (CMS'01) May 21–22, 2001, Darmstadt, Germany. Cham, Switzerland: Springer, 2001, pp. 197–211.
- [46] GitHub. Accessed: 2022. [Online]. Available: <https://github.com/qingzhenyun/qingzhenyunapi>
- [47] E. Trickel et al., "Toss a fault to your witcher: Applying grey-box coverage-guided mutational fuzzing to detect SQL and command injection vulnerabilities," in Proc. IEEE Symp. Secur. Privacy (SP), May 2023, pp. 2658–2675.
- [48] Post Policy—Amazon Simple Storage Service. Accessed: 2022. [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/API/sigv4HTTPPOSTConstructPolicy.html>
- [49] Response Header. Accessed: 2022. [Online]. Available: https://developer.mozilla.org/zh-CN/docs/Glossary/Response_header
- [50] Authenticating Requests: Browser-Based Uploads Using Post (AWS Signature Version 4). Accessed: 2022. [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-authentication-HTTPPOST.html>
- [51] Authenticating Requests: Using Query Parameters (AWS Signature Version 4). Accessed: 2022. [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-query-stringauth.html>
- [52] CVE-2017-11882. Accessed: 2022. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-11882>
- [53] CVE-2021-40444. Accessed: 2022. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-40444>
- [54] Object Storage Service (OBS)—Huawei Cloud. Accessed: 2022. [Online]. Available: <https://www.huaweicloud.com/intl/en-us/product/obs.html>
- [55] V. S. Sinha, D. Saha, P. Dhoolia, R. Padhye, and S. Mani, "Detecting and mitigating secret-key leaks in source code repositories," in Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories, May 2015, pp. 396–400.
- [56] Z. Yu Ding, B. Khakshoor, J. Paglierani, and M. Rajpal, "Sniffing for codebase secret leaks with known production secrets in industry," 2020, arXiv:2008.05997.



- [57] A. D. Diego, “Automatic extraction of API keys from Android applications,” Ph.D. dissertation, Dept. Eng., Universit’a degli Studi di Roma Tor Vergata, Rome, Italy, 2017.
- [58] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, “Hey, you, get o of my cloud: Exploring information leakage in third-party compute clouds,” in Proc. 16th ACM Conf. Comput. Commun. Secur., Nov. 2009, pp. 199–212.
- [59] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna, “Cloud strife: Mitigating the security risks of domain-validated certificates,” in Proc. Appl. Netw. Res. Workshop (ANRW). New York, NY, USA: ACM, 2018, p. 4, doi: 10.1145/3232755.3232859. [Online]. Available: <https://dl.acm.org/doi/10.1145/3232755.3232859>.
- [60] E. Pauley, R. Sheatsley, B. Hoak, Q. Burke, Y. Beugin, and P. McDaniel, “Measuring and mitigating the risk of IP reuse on public clouds,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2022, pp. 558–575.
- [61] X. Zhang, H.-T. Du, J.-Q. Chen, Y. Lin, and L.-J. Zeng, “Ensure data security in cloud storage,” in Proc. Int. Conf. Netw. Comput. Inf. Secur., vol. 1, May 2011, pp. 284–287.
- [62] A. Markandey, P. Dhamdhere, and Y. Gajmal, “Data access security in cloud computing: A review,” in Proc. Int. Conf. Comput., Power Commun. Technol. (GUCON), Sep. 2018, pp. 633–636.
- [63] D. Zhe, W. Qinghong, S. Naizheng, and Z. Yuhan, “Study on data security policy based on cloud storage,” in Proc. IEEE 3rd Int. Conf. Big Data Secur. Cloud (Bigdatasecurity) Int. Conf. High Perform. Smart Comput. (HPSC), IEEE Int. Conf. Intell. Data Secur. (IDS), May 2017, pp. 145–149.
- [64] Y. Zhang, C. Xu, and X. S. Shen, Data Security in Cloud Storage. Cham, Switzerland: Springer, 2020.
- [65] Z. Xia, T. Shi, N. N. Xiong, X. Sun, and B. Jeon, “A privacy-preserving handwritten signature verification method using combinational features and secure KNN,” IEEE Access, vol. 6, pp. 46695–46705, 2018.