International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 03 | March - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Securing Secret File Sharing Using Cyber Security

A.A. Nibe[1], Chaudhari Sakshi Satish[2], Palande Gauri Laxman[3], Vidhate Krupa Dilip[4], Kundhare Payal Gokul[5]

[1]Professor, Dept. of Computer Technology, P.Dr.V.V.P. Institute of Technology and Engineering, Loni, Maharashtra, India

[2,3,4,5] Final year Diploma Student, P.Dr.V.V.P. Institute of Technology and Engineering, Loni, Maharashtra, India

---

ABSTRACT - Various methods exist for preventing unauthorized access to data and networks. On an as-needed basis, firewalls are employed to secure passwords. On many occasions, they are insufficient. Thread is always monitoring systems and networks because of this. A computer system's intrusion detection system (IDS) can identify malicious online activity. Because of the tampering, hackers may be able to steal important and private files. However, it has been noted that the majority of intrusion detection systems and firewalls aim to safeguard computer systems from malicious outsiders. As both technology and modern societal norms develop, the needs for access control, a component of security, change. Numerous access control models originate in specific industries, such as healthcare and collaborative enterprises; however, in order to successfully implement the model in that specific usage environment, additional administrative measures, human factor considerations, and infringement management are still necessary. Using the media access control (MAC) address as a means of authentication and file protection is the subject of this project's detailed explanation.

Keywords: *File Access Control Mechanism, Device MAC address authentication.*

1.INTRODUCTION

Due to the dissolution of the traditional network boundary, protecting confidential file sharing is of the utmost importance in today's digital environment. Data that is particularly important to a company, such as proprietary information, financial records, and private communications, is increasingly vulnerable due to the widespread use of remote work and cloud storage. In order to execute corporate espionage or distribute ransomware, sophisticated attackers now focus on file-sharing protocols. The proliferation of "direct-to-cloud" upload models also brings new security holes that, if not adequately controlled, allow online users, servers, and cloud providers to evade security checks. Data breaches, legal responsibilities, and a drop in competitive advantage are all possible outcomes of insufficiently securing these channels.

In most cases, a mix of structural analysis and automated testing is used to find or secure a project. If we want to find bugs in the code, like incorrect file system permissions or faulty logic when dealing with file names, static analysis is the way to go. In dynamic testing, specialized tools are used to create malicious payloads, such as scripts or path traversal strings, and then the application is tested in real-time to see how it reacts to them. Furthermore, state-of-the-art projects leverage machine learning techniques like convolutional neural networks to detect file types from raw data patterns in the absence of metadata, and systematic state exploration to guarantee that file systems maintain consistency and security in the face of unforeseen system failures.

As a hardware-level security measure, MAC Address Authentication controls which devices can enter a network by checking their distinct Media enter Control address. When a network administrator creates a whitelist of allowed media access control (MAC) addresses, the system checks each device's hardware ID to make sure it's on the permitted list before giving access. Although this adds a first line of defense by limiting connections to known hardware, it is typically seen as an additional layer of protection due to the fact that malicious actors can "spoofed" or replicate MAC addresses. As a result, it works best when combined with additional forms of authentication, such as encryption or digital certificates.

[1] The author of the study by Kim et al. discovered four significant distinctions between the Linux and Windows file systems, which resulted in significant security vulnerabilities in the software, and Mr. Dong-uk conducted a comprehensive analysis of these differences. The author has also demonstrated that these variations are too complex for developers or compatibility layers to handle properly. Finally, the author proved that existing mitigation techniques don't work and that even experienced programmers struggle to fix these problems. Based on these findings, we can say that these vulnerabilities primarily originate from Windows file system design issues. The author argues that Microsoft should reconsider its file system architecture to make sure the Windows environment is stable.

[2] Recent comments by Yuanchao Chen et al. indicate that more and more websites are storing their files in the cloud. Users directly transferring data to cloud storage is one example. the author thoroughly examines the possible security concerns with this situation in this research. Obtaining and sending upload credentials, uploading and verifying data, and receiving and reacting to callback messages are the three most crucial procedures that the author examines in detail here. The case's data analysis uncovered six new vulnerabilities: unlimited upload credential acquisition (V1), a hole in the validity of uploaded credentials (V2), the ability to overwrite files (V4), the theft of files (V5), and callback notification spoofing (V6). Severe security breaches could affect web services, users, and cloud storage providers. The author investigates the practical consequences of these vulnerabilities by measuring the current popular websites that use public cloud services. It was shocking to everyone that every single one of the tested websites had some kind of security flaw. Although not all of the appropriate security teams or communities have replied positively, the author did inform them of the 79 new vulnerabilities found during the measurement. The authors are optimistic that their findings will contribute to future research on the security of cloud storage.

[3] Studying SEC 10-K filings, researchers Jonas Hielscher et al. uncovered information regarding SAT methods used by American companies. Since some companies may employ SAT in a clandestine manner, the results presented by the author should be regarded as a minimum. It is critical that this data is perceived as reliable because directors and corporations are individually liable for certifying that these filings are accurate and may face legal ramifications for any errors. At least 78.3% of companies utilize SAT, and 11.0 percent use MFA for their personnel, according to the author's statistics. Significant differences exist across industries, and there is a correlation between SAT and cybersecurity legislation. Similarly, larger companies are more likely to use SAT. Organizations often portray their employees negatively rather than addressing real security issues. The author's results pave the way for future SAT research grounded in evidence rather than estimates provided by entities with questionable motives, such as cybersecurity businesses or government bodies.

An examination of earlier research that was deemed a Litereature Survey is presented in the second part of this publication. Section 3 provides a comprehensive description of the proposed methodology, outlining the path of action. The experimental evaluation is covered in Part 4, possible modifications are discussed in Section 5, and the essay concludes with a conclusion on the existing plan.

2. LITERATURE SURVEY

[4] For the past decade, researchers Harun Oz et al. have examined security holes in practical applications that rely on file uploads. These vulnerabilities are on the rise, according to the author's findings, even if cybersecurity awareness and technology have both improved. It should be noted that according to the author's data, many file upload vulnerabilities are characterized by relatively easy assaults; yet, the most majority of these vulnerabilities have important implications for the availability, integrity, and security of systems. Strong security measures are urgently needed to protect file upload features on all platforms and communication services. The author's research aims to raise attention to this necessity.

[5] Using depth wise separable convolutional neural networks (CNNs), Muhamad Felemban et al. presented models for lightweight file fragment type categorization. Our goal is to create a model that outperforms the current state-of-the-art models in terms of inference time while maintaining accuracy. In particular, the author was able to write classification models with almost 100,000 parameters. Various situations were included in the FFT-75 dataset, which was used to evaluate the models. According to the evaluation results, the suggested models outperform the current state-of-the-art model for classifying file fragment types across all circumstances,

and they do it without significantly sacrificing accuracy. When dealing with classes that have a high rate of misclassification, there are a number of ways to improve the classification accuracy. The best way to come up with a design for data-specific models is to search for neural architectures [54], [55] when there are no hardware limitations. In addition, distillation in the network can remove superfluous connections for simple applications like JPEG classification compared to other file formats [56]. To further test their robustness and usefulness in real-world applications, the author plans to assess the performance of their models on imbalanced datasets in future work. This will ensure that the models can handle various and complex circumstances.

[6] A thorough investigation into the architectural flaws in the Windows file system that cause recurrent security vulnerabilities was provided by Dong-uk Kim et al. The author uncovered four major differences in the file systems of Windows and Linux, which lead to serious security flaws in actual software. The author also proved that developers or compatibility layers can't deal with these differences correctly. As a conclusion, the author demonstrated that current mitigation strategies are insufficient, and that even seasoned programmers have difficulty addressing these issues. In light of these results, the authors conclude that Windows file system design flaws are the primary source of these vulnerabilities.To ensure the Windows ecosystem is built on a secure foundation, we suggest that Microsoft rethink its file system architecture.

[7] Using NodeSec, a tool for analyzing Node.js applications and libraries in relation to UFU-related security issues, Harun Oz et al. examined the file upload ecosystem in Node.js. Their findings included eleven widely-used Node.js web applications, nineteen critical vulnerability reports (CVEs), and security flaws in the file upload libraries. The authors of this study hope that by drawing attention to the need for stronger security measures in online apps and libraries that allow users to upload files, they will help pave the way for more effective methods of preventing UFU assaults.

[8] A novel method for efficient file system consistency testing was introduced by Jianzhong Liu et al. as SnapCC. To go farther into finding consistency issues in contemporary file systems, SnapCC suggests using systematic state exploration, which is different from earlier efforts. With the help of a system-call-based File System Invocation Sequence Generator, a Systematic On-Disk State Explorer, and an Automatic Valid State Finder, SnapCC is able to automatically determine all valid file system states and systematically find all possible on-disk states while a file system invocation sequence is being executed. Quickly and accurately, SnapCC discovers consistency flaws in file systems by comparing the two sets of entries and recognizing any invalid state. The author's studies showcase SnapCC's potential, revealing a 16% to 44% increase in coverage statistics compared to Hydra. Additionally, 15 new crash-consistency flaws were identified in popular file systems such as BTRFS, F2FS, and ext4. SnapCC's adaptability to file systems was demonstrated when the author discovered seven new crash consistency flaws in different Linux file systems, including UFS, XFS, BCacheFS, and OpenZFS. By systematically exploring on-disk states and generating invocation sequences, the author demonstrated that SnapCC is effective and can trigger more states in the code of the file system.

[9] In order to address a wide range of configuration challenges, Tabassum Mahmud et al. developed an extensible framework named ConfD and published a study on 78 real-world configuration issues. By utilizing configuration dependencies, the author's tests on Ext4, XFS, ZFS, and WiredTiger show that ConfD can effectively handle configuration difficulties. The author plans to continue working on ConfD and exploring additional systems, as mentioned in Section 6, in the near future. In order to tackle the growing problem of configurations in general, the authors are hoping that ConfD can pave the way for further study.

[10] The unique ByteRCNN machine learning model for file fragment type recognition is described by Kristian Skračić et al. and is built using byte embeddings in addition to convolutional and recurrent elements. While using more parameters and running slower than some state-of-the-art models, the authors show that ByteRCNN achieves classification accuracy that is comparable to or even better than state-of-the-art models using training and evaluation on the FiFTy scenario #1 dataset. The authors achieved an average accuracy of 71.1% on 512-byte fragments and an average accuracy of 83.9% on 4,096-byte fragments [1], [19], [20]. It is worth mentioning that the author accomplished this finding by utilizing a single model in several contexts, in contrast to other pertinent studies that utilized distinct models. However, when tested on different datasets, ByteRCNN showed that there is much space for improvement in terms of classification accuracy when it comes to open-set and closed-set recognition and file fragment type identification. Since there are currently no

publicly available state-of-the-art classification models, the author backs up this claim by comparing the FiFTy scenario #1 model to the same datasets and finding that it achieves much lower classification results on datasets [2, 3], [4, 5] than on the FiFTy scenario #1 dataset. It appears that categorization in both closed-set and open-set scenarios is tough for fragments originating from media files with different codec and compression settings. Based on the findings, new lines of inquiry may be warranted. Given the abundance of literature on effective file fragment type identification models, it would be remiss not to provide the implementations of these models. If they were to publish their implementations, we could see how well they performed in each class, see where they fell short, and maybe even use them simultaneously to improve our classification results. Future studies should additionally focus on open-set recognition in order to accommodate for more difficult but realistic file fragment type identification scenarios. Last but not least, existing or future datasets that contain bits from various files of the same type should be the focus of more research. The author is hopeful that ByteRCNN and the findings presented here will guide future studies in this area.

[11] A distributed file system (DFS) developed for consortium systems was presented by Yitong Cheng et al. DeFS strikes a compromise between decentralization and functionality via its two-layer design, which includes a data layer with many public nodes and a metadata layer with a small number of consortium nodes. To ensure data availability, the big data layer stores blocks of encrypted data. In order to offer flexible access control and complex functionality, the metadata layer keeps track of file metadata. In order to facilitate data-layer characteristicaware block routing, a Multi-Ring DHT (MR-DHT) protocol was developed. Management of metadata among dangerous metadata nodes is the goal of a Trusted Consistent Hashing (TCH) protocol. Additionally, the L-Cache approach is employed to expedite the block routing process. The authors have tested a DeFS prototype on a large network spanning multiple regions and containing more than 1,500 nodes. According to the findings, DeFS outperforms IPFS in terms of throughput by a factor of up to 15.57.

3. METHODOLOGY

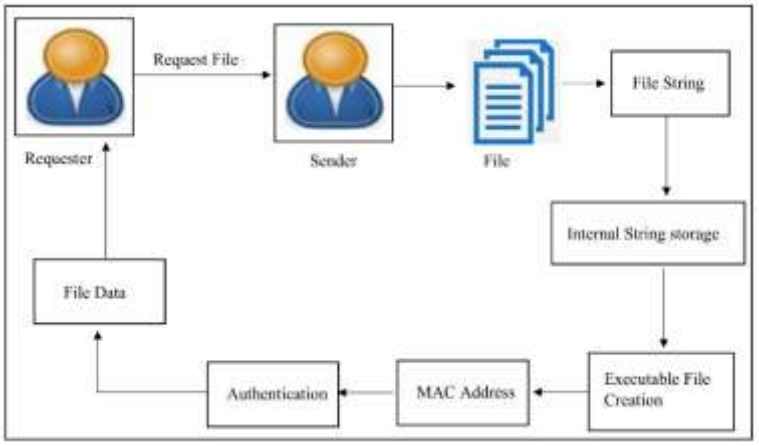

Figure 1: Proposed model System Overview

Step 1: File Content Extraction:

The system starts when the sender creates a login ID by giving many attributes, including an email address, a password, and a username. Once the user has registered and received their username and password, they will be able to access the system. Following successful login, the user can choose to share several files with the recipient. The files are effectively sent to a storage facility and made available to the server. After that, the recipient can go to the server housing the files and ask for a specific file that was requested by the sender. To prepare for the following step, the content is read into a string and then stored.

Step 2: Internal File Storage:

The receiver takes the request in str form and extracts the file's content when the sender sends it. The creation of a protected file makes efficient use of strings. You can input text files (docs) and PDFs (pdfs) into the system. You can access these files through their corresponding jar files, which allow you to interface them with the Java code. When the finalist converter's material is used in the following step of creating executable files.

Step 3: Executable File Creation and MAC Address Authentication:

The data is stored and incorporated into Java code once the file content is extracted. When the whole file is processed, the contents of the file can be seen in a text field. The clean and build option of the net beans IDE cannot convert this Java code to a JAR file. Now that the JAR file is ready, the database will show it as secured, and the user can receive the file. The file is transmitted to the recipient's IP address using the socket application. By using the receiving device's Mac address, the file is

successfully authenticated and authorized before being sent. This makes sure that the file can only be opened on the specific device associated with the Mac address. Whenever the JAR file is extracted, it checks the device's Mac address to see if it matches the one in the file. Only then is the file content authorized to be seen.

4. RESULTS AND DISCUSSIONS

The suggested protocol for a reliable Secret File Sharing system was developed using the Netbeans IDE and a number of Java tools. Because of this, the secure file might be generated automatically in a real-time situation. A Windows laptop with an Intel Core i5 processor, 8 GB of RAM, and other developer-specific hardware was used to execute the recommended method. Examining the mistakes made during deployment and execution is essential for determining the solution's reliability.

Performance Evaluation through Root Mean Square Error (RMSE)

To find the root mean square error (RMSE), one subtracts the error from the difference between the predicted and obtained evaluations. Because of this, a practical policy that can reliably assess the method's performance is within reach. The correct assessment of the flood warning is investigated and its root-mean-squared error (RMSE) is computed by using equation 1 below.

RMSE= √((xp-xo)) 2 ---------- (1)

Where

Xp – Expected number of Proper Secure file Generation

Xo – Obtained number of Secure file

No of Experiments	Number of Files	Obtained No of Secure files	MAE
1	5	5	0
2	8	8	0
3	12	11	1
4	14	13	1
5	17	17	0

Table 1: Obtained Results for Mean Absolute Error

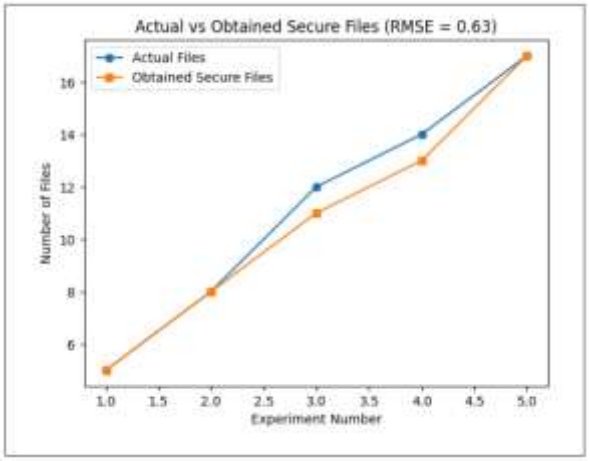

Figure 3: Graphical Representation of RMSE Results

The outcomes of five trials that contrasted the real file count with the calculated secure file count are shown in table1. If the predicted and actual values are different for each experiment, then the mistake is that experiment. There is no error in the majority of the trials, however there is a single file error in experiments 3 and 4. The RMSE, or Root Mean Square Error, is calculated to measure how accurate the predictions are. By averaging the squared errors, we get a Mean Square Error (MSE) of 0.4; taking the square root of this number yields an RMSE of about 0.63, suggesting that the disparity between the predicted and actual values is negligible.
A comparison of the obtained number of secure files with the actual number of files across five studies is shown in figure 3, a graph. Experiments show that the method is very accurate, since the number of secure files obtained is very near to the actual number of files. Experiments 3 and 4 show small discrepancies, with values produced that varied by one file. Based on these discrepancies, we can determine an RMSE of 0.63, which is extremely small and indicates that the system's predictions are spot on and in good agreement with the data.

5. CONCLUSION AND FUTURESCOPE

In conclusion, in today's digital age, companies must prioritize the security of secret file sharing due to the fact that the attack surface has been greatly expanded by the trend toward remote work and cloud-integrated infrastructures. It is feasible to mitigate risks even when file metadata is missing or corrupted by embracing proactive identification techniques like static analysis for finding design weaknesses and machine learning models like ByteRCNN for fragment classification. This goes beyond traditional perimeter defenses. While MAC address verification has its limits when it comes to spoofing, it is nevertheless an important authentication

method to adopt since it adds a layer of hardware-level gatekeeping that improves security.

This project proves that decentralized architectures and AI-driven automated testing frameworks are the way to go for safe file sharing in the future. These frameworks can automatically adjust to new threats as they emerge. Future developments can make sure data is consistent across different platforms by fixing basic file system inconsistencies and configuration dependencies. This would reduce single points of failure. With the increasing complexity of cloud ecosystems, it will be crucial to implement a zero-trust strategy that integrates these advanced identification methods with ongoing multi-factor verification in order to safeguard shared data.

In the future, due to the fact that remote work and new cloud infrastructures have eliminated the conventional network boundary, leaving sensitive data vulnerable to sophisticated attacks such as industrial espionage and ransomware, it is crucial to secure secret file sharing. This project shows that organizations can greatly decrease their vulnerability to data breaches and system inconsistencies by combining proactive identification techniques with hardware-level defenses, such as MAC address authentication. These techniques include static code analysis and machine learning for file fragment classification. In an increasingly linked and dispersed digital world, data integrity and confidentiality must be guaranteed by adopting a security-by-design strategy.

Integrating autonomous, AI-driven security frameworks that can detect and fix file system vulnerabilities in real-time is the future scope of this research. To guarantee data immutability and remove single points of failure, future research will investigate blockchain-based and decentralized storage solutions, like consortium systems. In addition, by swapping out static hardware identifiers for ongoing, multi-factor verification of user behavior, device health, and geographic context, zero-trust architectures will offer a more resilient defense.

REFERENCES

[1] D. Kim, J. Park, S. Oh, H. Kim, and I. Yun, "Windows plays Jenga: Uncovering Design Weaknesses in Windows File System Security," in Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), Taipei, Oct. 13-17, 2025, pp. 3900-3914. [Online]. Available: https://doi.org/10.1145/3719027.3765217.

[2] Y. Chen, Y. Li, Y. Lu, Z. Pan, Y. Chen, S. Ji, Y. Chen, Y. Li, and Y. Shen, "Understanding the Security Risks of Websites Using Cloud Storage for Direct User File Uploads," IEEE Transactions on Information Forensics and Security, vol. 20, pp. 2677–2692, 2025, doi: 10.1109/TIFS.2025.3544082.

[3] J. Hielscher and M. Golla, "Quantifying Security Training in Organizations Through the Analysis of U.S. SEC 10-K Filings," in Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), Taipei, Taiwan, Oct. 13-17, 2025, pp. 51-65, doi: 10.1145/3719027.3765179.

[4] H. Oz, G. S. Tuncay, A. Aris, A. Kharraz, and S. Uluagac, "Beyond the Upload Button: A 10-Year Retrospective on Security Issues Within File Upload," IEEE Communications Magazine, vol. 63, no. 2, pp. 104–110, Feb. 2025, doi: 10.1109/MCOM.0012400253.

[5] M. Felemban, M. Ghaleb, K. Saaim, S. Alsaleh, and A. Almulhem, "File Fragment Type Classification Using Light-Weight Convolutional Neural Networks," IEEE Access, vol. 12, pp. 157179-157191, 2024, doi: 10.1109/ACCESS.2024.3486180.

[6] Y. Chen et al., "Understanding the Security Risks of Websites Using Cloud Storage for Direct User File Uploads," IEEE Transactions on Information Forensics and Security, vol. 20, pp. 2677–2692, 2025, doi: 10.1109/TIFS.2025.3544082..

[7] H. Oz, A. Acar, A. Aris, G. S. Tuncay, A. Kharraz, and S. Uluagac, "(In)Security of File Uploads in Node.js," in Proceedings of the ACM Web Conference 2024 (WWW '24), Singapore, May 13-17, 2024, pp. 1573–1584, doi: 10.1145/3589334.3645342.

[8] J. Liu, Y. Shen, Y. Xu, H. Sun, and Y. Jiang, "SNAPCC: Effective File System Consistency Testing Using Systematic State Exploration," ACM Transactions on Storage, vol. 19, no. 4, pp. 1–28, 2023, doi: 10.1145/3614436.

[9] T. Mahmud, O. R. Gatla, D. Zhang, C. Love, R. Bumann, V. Girimaji, and M. Zheng, "Analyzing Configuration Dependencies of File Systems," ACM Transactions on Computer Systems, vol. 43, no. 4, art. no. 14, pp. 1–28, Nov. 2025, doi: 10.1145/3747177.

[10] K. Skračić, J. Petrović, and P. Pale, "ByteRCNN: Enhancing File Fragment Type Identification With Recurrent and Convolutional Neural Networks," IEEE Access, vol. 11, pp. 138175-138187, 2023, doi: 10.1109/ACCESS.2023.3340441.
[11] Y. Cheng, Y. Yu, and B. Chen, "DeFS: A Decentralized and High-Performance File System for Consortium Systems," in Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), Taipei, Taiwan, Oct. 13–17, 2025, pp. 2960–2974, doi: 10.1145/3719027.3765123.