Redesign and Feasibility Verification of Access Control System Based on Correlation Among Files

Yuki Kodaka
Department of Informatics, The Graduate University for Advanced Studies
Tokyo, Japan
email: y_kodaka@nii.ac.jp

Hirokazu Hasegawa
Center for Strategic Cyber Resilience Research & Development, National Institute of Informatics
Tokyo, Japan
email: hasegawa@nii.ac.jp

Hiroki Takakura
Center for Strategic Cyber Resilience Research & Development, National Institute of Informatics
Tokyo, Japan
email: takakura@nii.ac.jp

***Abstract*—File access control is an effective method for protecting information from unauthorized access both inside and outside an organization. However, conventional methods based on organizational structure have some limitations. Modern business requires flexible access control that reflects the dynamic changes in workflow. Still, it is difficult to achieve the requirement at the same time the prevention of information leakage and destruction due to cyberattacks. Therefore, this paper proposes an access control system based on the correlation among files. The correlation is inferred from users' access histories within the same group, and access privilege is determined based on the strength of the correlation. This system adapts to changing access needs and prevents unauthorized access by automatically denying access with low file-to-file correlation in a series of accesses. The initial implementation of the system was carried out in a simplified environment, which raised issues about whether the system could be feasible and efficient in real-world, more complex scenarios. This work extends the findings of our previous paper by addressing identified issues with the proposed system through targeted modifications. To further validate the system's performance and feasibility in real-world scenarios, we conducted subsequent implementation and verification experiments under conditions that were not only more practical but also involved higher loads. These efforts aimed to rigorously test the system's scalability and efficiency in environments that closely mimic actual operational conditions. As a result of these modifications and experiments, the system demonstrated the capability to handle high-load conditions efficiently. This outcome suggests that the potential impact on file system processing due to the introduction of new features via the proposed system is not serious. Therefore, our extended research confirms the proposed system's robustness and suitability for real-world application, highlighting its ability to maintain efficiency even under significant stress. To ensure the feasibility of the proposed system, future work should address the effectiveness issue.**

***Index Terms*—*File Access Control*; *Graph Theory*; *Bell-LaPadula Model*.**

I. Introduction

This paper follows up our previous paper "Design and Implementation of Access Control Method Based on Correlation Among Files" already published in the proceedings of CENTRIC 2023 [1].

File access control has long been used as an effective method of protecting an organization's information assets. It prevents unauthorized accesses by users and minimizes information leakage due to cyber attacks. Various access control methods have been proposed and developed [2]–[4].

However, many of the current methods and operations are not flexible enough. Due to changes in the situations, access control loses accuracy over time [5]. In some cases, policymakers (high-level policy architects) and implementers of policy designed by others are separated. And policies are often managed by several persons rather than a single person [6]. These also make flexible operation difficult.

Strict access control is required, especially in environments where sensitive information is handled. For example, the Bell-LaPadula model [7] was proposed to prevent the leakage of information known only to the supervisor to subordinates. However, in many cases, supervisors can write to files that their subordinates can read and write to.

According to Proofpoint report [8], the cost of insider threats has surged from \$8.30 million in 2018 to \$15.38 million in 2022, an 85% increase. In order to mitigate insider threats, not only technical approaches like access control systems, but also non-technical approaches like user behavior analytics are needed [9].

To address these issues, we point out two challenges. First, the principle of least privileges frequently discussed, but there is no practical method of assigning access privileges based on it. As business progresses, the required access privileges are constantly changing. However, such changes are always incurred and unpredictable beforehand. Second, there is no method of implementing granular file access control in response to constantly changing needs. Obviously, the assignment of access privileges should be done with caution. Excessive access privileges increase the risk of leakage or destruction. On the other hand, insufficient access privilege affects the ability of users to perform their operations. As a result, it may undermine the efficiency and productivity of the organization.

To solve these problems, we have proposed an access control system based on the correlation among files [10]. The correlation is inferred from user's access histories. The system automatically determines whether access is allowed or denied based on the degree of the correlation. It responds to access needs based on changing situations. The system automatically denies accesses with low correlation. It prevents excessive expansion of the access privilege. It is assumed that access by malware is an uncorrelated access. Or, even access by an insider is assumed to be uncorrelated if it is not related to the person's business. These accesses are different from legitimate users. This system can prevent such file accesses. We designed and implemented our system to include file-correlation-based access control as one of its elements. However, there were issues about this design, and the implementation was in a simplified environment. To verify the system's feasibility, it was necessary to address these issues and verify its scalability in a more practical environment. For this study, after making modifications to our system, we extended implementation and verification experiments under more practical and high-load conditions.

This paper is organized in the following sections. Section II refers to related work to this paper. Section III describes the assumptions of the proposed system. After that, we explain the design of the proposed system. Section IV describes the implementation of the proposed system. Section V describes the evaluation experiment of the proposed system. Section VI concludes this paper and presents future work.

II. Related work

Users are sometimes denied access to files they actually have the right to use, and administrators are required to modify the access control of the files. They might make a misconfiguration at the modification that gives more access privileges than necessary. Xu et al. investigated how and why such problems occur [11]. Although several reasons for misconfiguration are shown, administrators must solve such problems by themselves, and the possibility of misconfiguration and the burden on administrators remains.

Beckerle and Martucci proposed the metrics to evaluate and quantify access control rule sets in terms of security and usability [12]. The metrics helps users generate better rule sets. One of the evaluation indicators is the difference between the owner's intention and the rule set. However, the actual method of getting the intention is out of the scope of the paper.

Mazurek et al. proposed reactive policy creation in response to user's access request [13]. The experiment involves sharing files on digital devices at home with people, including supervisors and co-workers. If a user tries to access a resource but lacks sufficient privilege, they can use the proposed system to send a request to the resource owner, who can opt to update their policy and allow the access. This method requires the file owner to make determinations for all unauthorized access.

Shalev et al. proposed an improved method for containers that allows monitoring and logging of operations by the system administrator [14]. The operations used by system administrators include not only support by internal IT department employees, but also by third parties such as storage service providers and automated management tools used by the IT department. The system administrator is expected to operate based on user requests (tickets in this paper), but there is no mention of whether or not those requests are necessary.

Desmedt and Shaghaghi proposed an access control method that considers three dimensions: subject, object, and operation, rather than the conventional two dimensions of subject and object [15]. These mainly counter internal threats and provide granular access control by controlling operations. It shows how to implement granular access control, but does not mention how to update access privileges once they have been set.

Although not access control, research has been conducted to identify legitimate and malicious users based on user behavior. Mannila et al. introduced the Window-based Episode Discovery (WINEPI) and Minimal Occurrence-based Episode Discovery (MINEPI) algorithms, offering advanced methods for analyzing patterns in event sequences known as 'episodes' [16]. WINEPI calculates the frequency of episodes within time windows, effectively identifying common patterns. MINEPI complements this by focusing on the minimal occurrences of episodes, revealing the shortest intervals for episode beginnings. These approaches provide a comprehensive framework for pattern detection, crucial in fields such as telecommunications and user behavior analysis.

Camina et al. focused on masquerade detection using user navigation structure, employing the Naïve Bayes classifier [17]. The 'user navigation structure' refers to how a user interacts with their file system. In their 2014 research, they shifted focus to abstracting user behavior into tasks, utilizing both Naïve Bayes and Markov Chain classifiers [18]. Here, a 'task' is associated with a file system (FS) directory, encompassing a number of related file system objects. These studies illustrate the evolution in detecting masquerades, from analyzing direct navigation patterns to abstracting user actions into categorized tasks.

Huang et al. addressed the challenge of intrusion detection through the analysis of user file access patterns [19]. Their approach involves modeling user behavior using both Non-frequency-based and Frequency-based feature sets. The former includes metrics like the duration and diversity of file access paths, while the latter focuses on aspects like access frequency and file type variations. This comprehensive analysis of user behavior significantly contributes to the effectiveness of intrusion detection systems.

Mehnaz and Bertino explored anomaly detection in file system accesses by leveraging enhanced user profiles [20]. These profiles, incorporating frequency and access cluster data similar to Mannila et al.'s approach [16], are further enriched by block-level analysis derived from access size logs. The research involves comparing these comprehensive profiles against users' access patterns to effectively identify anomalies, providing an advanced framework for detecting irregularities in file system usage.

In research that utilizes these access patterns for detecting unauthorized access, it is possible to distinguish between the main similar access patterns and others. However, there is a risk of misjudging the unique access patterns of individual users.

III. Design of Proposed System

In this section, we first discuss the file correlation-based access control system proposed in our previous works [1], [10]. We then address the issues identified with the system's design. Finally, we describe the modifications made to the design in response to these issues.

A. *Assumption*

The proposed system assumes an organization consisting of a hierarchical structure as shown in Fig. 1. Although agile development is practiced, a certain organizational structure still exists. This paper calls the largest segment of an organization, such as a department in a typical enterprise, a group. Divided units within the group are called subgroups, and further divided units within a subgroup are called subsubgroups. In the example shown in Fig. 1, each department is a group, and each section is a subgroup.

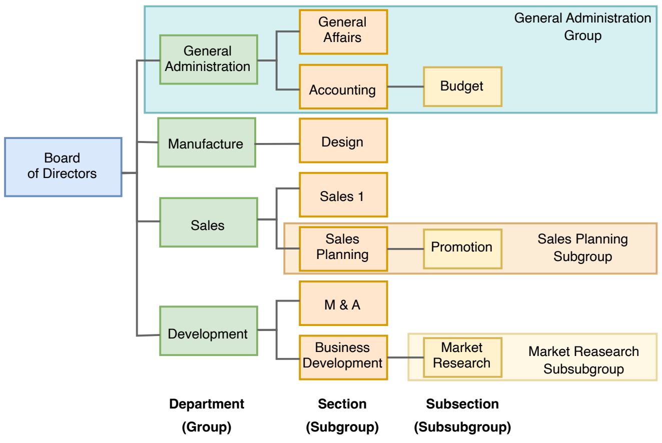

Fig. 1: Example of Organizational Structure

In this paper, we propose a file access control system based on the correlation among files. Our working hypothesis is that user access patterns are not random or irregular, but rather exhibit a certain level of consistency and commonality.

In collaborative environments, it is a recognized fact that users tend to be team-oriented [21]. This study operates under the hypothesis that typical users within a Collaborative Information System are likely to form and function as communities.

Therefore, we assume that the proposed system will be effective in enhancing the security of team-based workflows. This approach is particularly well-suited for business processes that are team-oriented.

Fig. 2 shows the assumed access control environment. Generally, an Access Control List (ACL) is implemented with coarse-grained, such as per folder, for groups or subgroups. The access privilege under a folder is determined using the information in a directory service, such as Active Directory (AD). If fine-grained access control is to be implemented, it is set by the file owner or the administrator, but their load becomes significant.

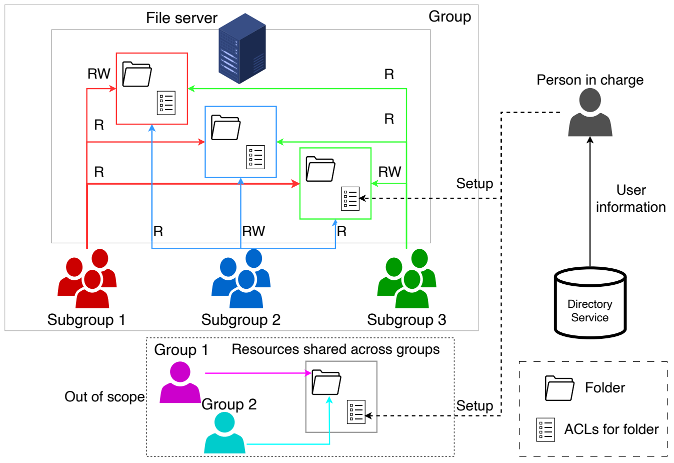

Fig. 2: Assumed Access Control Environment

In this paper, resources shared within each group are targeted, and resources shared across groups are out of scope.

B. *Overview of Proposed System*

With the dynamic changes in the environment, the required privileges are constantly changing. To respond to such changes, the proposed system has the capability to continuously evaluate and manage access privileges. It executes two main functions: the addition of necessary access privileges and the revocation of unnecessary ones.

1) Addition of necessary access privileges: Granting of access privileges is executed on the basis of two-step determination, i.e., automatic determination and manual determination. When a user tries to access a file without access privileges, such access is denied first. The proposed system triggered by the event of access denial. Then the proposed system performs an automatic access determination on the denied access. If the results of the determination show that there is a correlation between the files to which the user has access privilege and the denied files, the ACL is changed to "allowed" to access the file. Even if the access is denied as a result of the automatic access determination, the user can request a manual access control determination if the access is truly necessary. In this paper, it is assumed that manual determination is performed by the file owner.

2) Deletion of unnecessary access privileges: The revocation is executed when there has been no access for a certain period of time. The proposed system records the file access history of each user for a certain period in the past. If a user has not accessed a file for this period, the access privilege is considered no longer needed, and it is revoked. The longer the period of access history recorded, the longer access privileges can be retained, which offers advantages from a usability

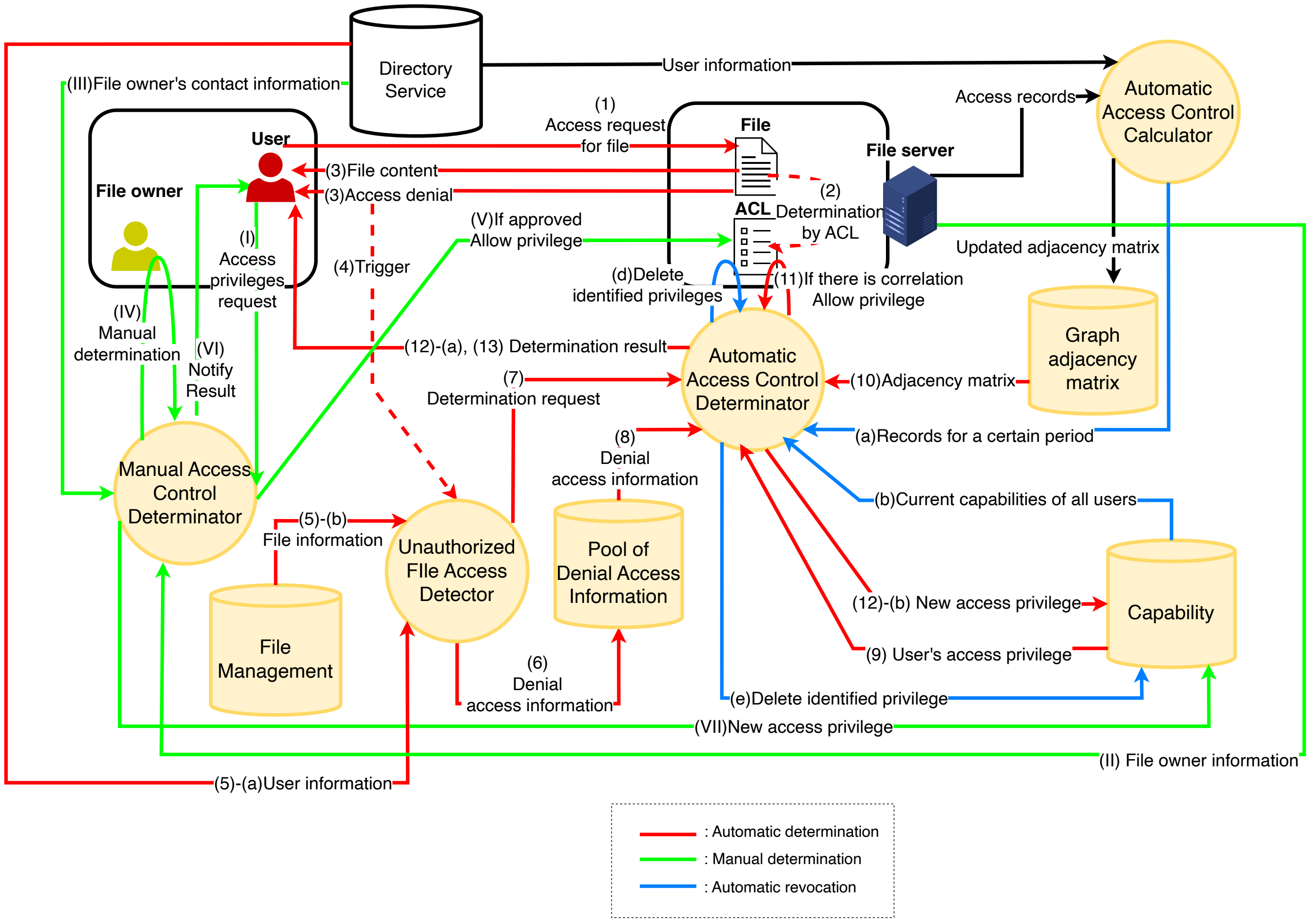
Directory Service
User information
Automatic Access Control Calculator
Access records
(III)File owner's contact information
(1) Access request for file
File
File server
User
File owner
(3)File content
(3)Access denial
(2) Determination by ACL
ACL
(I) Access privileges request
(4)Trigger
(V)If approved Allow privilege
Updated adjacency matrix
(d)Delete identified privileges
(11)If there is correlation Allow privilege
(IV) Manual determination
(VI) Notify Result
(12)-(a), (13) Determination result
Graph adjacency matrix
Automatic Access Control Determinator
(7) Determination request
(10)Adjacency matrix
(8) Denial access information
(a)Records for a certain period
Manual Access Control Determinator
(5)-(b) File information
Unauthorized FIle Access Detector
Pool of Denial Access Information
(b)Current capabilities of all users
File Management
(6) Denial access information
(12)-(b) New access privilege
Capability
(9) User's access privilege
(e)Delete identified privilege
(VII)New access privilege
(II) File owner information
(5)-(a)User information
: Automatic determination
: Manual determination
: Automatic revocation

Fig. 3: Architecture of Proposed System (yellow)

perspective. On the other hand, the shorter the recording period, the quicker unused access privileges are removed, providing benefits from a security perspective. Given these trade-offs, the period of recorded access history should be flexibly set according to the organization's needs and the job operations. As an example, this paper records the file access history for the past month (the past 30 days).

C. Architecture of Original System

Fig. 3 shows the architecture of the proposed system. The compoment of the proposed system is yellow color. It is assumed that the other components, apart from the proposed system, already exist within the organization. It includes the assumed flow of access determination and the source of information necessary for the determination. The proposed system consists of an automatic access control calculator (AACC), unauthorized file access detector (UFAD), automatic access control determinator (AACD), manual access control determinator (MACD), and four databases store the target file information, the denied access information, the adjacency matrix of the graph, and the access privileges by each user.

The procedure of the proposed system process is as follows. The term "username" refers to a unique identifier for users, similar to a user ID, which is managed within a directory service. Similarly, "filename" refers to a unique identifier that can be assigned to a full path or an ID given to a file.

- Automatic determination of access privileges (red line)

(1) Users access files
(2) File server determines whether the access is allowed or denied based on the ACL set for each file
(3) (a) If allowed, the user gets the file content
    (b) If denied, the user is notified of access denial, and the access denial is recorded in the access log
(4) The UFAD is triggered by the event of access denial
(5) (a) The UFAD fetches the target user information from the database for directory services
    The user information is username, rank, affiliation
    (b) The UFAD fetches the target file name from the database for file management database
(6) If the record shows denial access for the target files and the users,
    the UFAD store it in the pool of denial access information database
    The denial access information is Timestamp, Username, Filename, Accesstype (R or W)

(7) The UFAD makes a determination request to the AACD
(8) The AACD fetches the denial access information from the pool of denial access information database
(9) The AACD fetches the user's access privileges from the the capability database
(10) The AACD performs the access determination using graphs
(11) If there is a correlation, The AACD allows the access privilege to the user
(12) If allowed,
   (a) the AACD notifies the user of the result 'Allowed' The notification includes Timestamp, Username, Filename and Accesstype (R or W)
   (b) The AACD adds the new access privilege to the capability database
(13) If denied,
the AACD notifies the user of the result 'Denied'
The notification includes Timestamp, Username, Filename and Accesstype (R or W)
Additionally, the user has the option to submit a new access privilege request along with the reason for the request

- Manual determination of access privileges (green line)

(I) The MACD receives the new access privileges request from users
(II) The MACD fetches the file owner information from the File server
(III) The MACD fetches the file owner's contact information from the directory service
The contact information is like email address or chat-tool used in the organization
(IV) The MACD requests the file owner to determine whether the access is allowed or not
(V) If approved, the MACD allow the access privilege to the user
(VI) The MACD notifies the result to the user
(VII) The MACD adds the new access privilege to the capability database

- Deletion of unnecessary access privileges (blue line)

(a) The AACD fetches the records for certain period from the AACC
(b) The AACD fetches the capabilities, which detail the current access privileges of each user
(c) The AACD compares the records with the capabilities identifying user names, file names, and access privileges (R or W) that are present in the capability database but not in the records
(d) The AACD deletes the identified privileges from the ACL
(e) The AACD also deletes the identified privileges from the capability database

D. Function of each Component

The functions of each component are described below.

1) Automatic Access Control Calculator (AACC): The AACC calculates graphs utilizing graph theory for correlation determination. The graph infers the correlation among files based on the user's access histories, which are sourced as records from access logs on the file server. Each event of a user accessing a file is logged as a record.

The graph calculation is planned to schedule at two specific times: the change of the date and at arbitrary intervals during business hours. Firstly, at the stroke of midnight, the proposed system fetches access records from the past 30 days up to the day before from the file server, along with usernames, ranks, affiliations as the necessary user information from the directory service. This data is used to calculate the graph's adjacency matrix, as indicated by the black lines in Fig. 3. During business hours, the proposed system continues to update this information at set intervals, such as hourly. It fetches the current day's access records and user information to update the adjacency matrix. The final graph is a composite of the matrices generated at the date change and during the business hours. This composite graph is then stored in a database for adjacency matrices, as indicated by the black lines in Fig. 3.

For the graph calculation, the AACC maintains a dataset that combines the access records from the past 30 days (up to the day before) obtained at the moment the date changes, with the access records of the current day acquired during business hours. This ensures that the AACC possesses continuous records of access from 30 days prior up to the most recent data available at the predetermined intervals.

The calculation procedure is as follows. Data is access records for a certain period, which is the past month (the past 30 days) in this paper.

a) Extract specific information from access records: Specific information in the access records is fetched as the access history used in the calculation. The specific information is "Timestamp", "Accesstype" (R or R/W), "Username", "Filename". The extracted access histories are sorted by username and time.

b) Categorize access histories by user rank and access type: Extracted access histories are categorized by user rank and access type. Ranks are assumed to be hierarchical. For example, from the top, director, manager, section chief, member. For each user rank, two access histories are categorized. One is the access history of the "Read" access type for users below the same rank. The other is the access history of the "Write" access type for users in the same rank.

c) Create graphs from access histories: An example graph is shown in Fig. 4. The graph consists of nodes (V_1, V_2, V_3) and links between nodes (L_{1-2}, L_{2-3}, L_{1-3}). In the graph, nodes represent files. Links represent the correlations among files. The graph is assumed to be undirected. The order of accesses, A-B and B-A are counted as the same.

The graph is calculated using an adjacency matrix. Adjacency means that node i and node j are adjacent to link $i-j$ in the graph. An adjacency matrix is a square matrix used to represent a finite graph. The elements of this matrix indicate if a pair of nodes is adjacent or not in the graph. If so, it

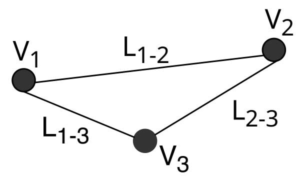

Fig. 4: Example of Graph

indicates the weights of the links between adjacent nodes. An example of an adjacency matrix is shown in Table I.

TABLE I: EXAMPLE OF ADJACENCY MATRIX

$$\begin{array}{c|cccc} & \text{FileA} & \text{FileB} & \text{FileC} & \text{FileD} \\ \text{FileA} & 0 & 3 & 0 & 1 \\ \text{FileB} & 3 & 0 & 1 & 5 \\ \text{FileC} & 0 & 1 & 0 & 1 \\ \text{FileD} & 1 & 5 & 1 & 0 \end{array}$$

The following procedure is used to calculate link weights.

a. Get a list of files by rank and access type from categorized access histories
b. Determine the size of the adjacency matrix from the list
c. Create adjacency matrix initialized to 0
d. Calculate weights of links from categorized access histories

   Add link weights between consecutive files in the categorized access histories, if the same user accesses different files within a certain period of time (one hour in this case). Furthermore, when calculating weights, we consider the time inclination shown in (1) based on the timestamp:

$$1 - \left(\frac{D}{D_{\text{max}}}\right)^n \qquad (1)$$

   where D is the number of days elapsed from the most recent day, D_{max} is the number of calculation days, and n is an adjustment parameter.
e. Normalize weights of links

   Let A be the adjacency matrix before normalization, and $S(n)$ be the total weight of the links connected to each node n. Normalization is performed as shown in (2):

$$B(i,j) = \frac{A(i,j)}{S(i)} + \frac{A(j,i)}{S(j)} \qquad (2)$$

   where $B(i,j)$ is the element at the ith row and jth column of the adjacency matrix after normalization, $A(i,j)$ is the element at the ith row and jth column of the adjacency matrix before normalization, and $A(j,i)$ is the element at the jth row and ith column of the adjacency matrix before normalization. Also, round off to the second decimal place.

The algorithm, which is the pseudocode of the above procedure, is shown in Algorithm 1.

Table II shows the results of the above calculation steps using Table I as an example.

Algorithm 1
Calculating link weight based on access histories with time inclination

```
Require: Categorized access histories, list of target users, list of target files
Ensure: Adjacency matrices for each rank and access type
1: Define list of ranks and list of access types
2: for each rank in list of ranks do
3:     for each access type in list of access types do
4:         Load a access history data
5:          from categorized access histories
6:         Obtain list of unique filenames from data
7:          and sort them as specified
8:         Initialize adjacency matrix
9:          based on sorted filenames
10:        for each user in data do
11:            Group access histories by user
12:            for each history in user's group do
13:                Set time limit according to access type
14:                 (1 hour for read, 2 hours for write)
15:                if time difference to a next history
16:                 within limit and
17:                 a next history refers to different file then
18:                    Calculate elapsed time
19:                     since the log's timestamp
20:                    Calculate weight
21:                     using time decay formula:
22:                        $weight = 1 - (\frac{\text{elapsed days}}{D_{\text{max}}})^n$
23:                    Add calculated weight
24:                     to corresponding elements
25:                     in adjacency matrix
26:                end if
27:            end for
28:        end for
29:        Output adjacency matrix
30:    end for
31: end for
```

TABLE II: EXAMPLE OF NOMALIZED ADJACENCY MATRIX

$$\begin{array}{c|cccc} & \text{FileA} & \text{FileB} & \text{FileC} & \text{FileD} \\ \text{FileA} & 0.00 & 1.08 & 0.00 & 0.39 \\ \text{FileB} & 1.08 & 0.00 & 0.61 & 1.27 \\ \text{FileC} & 0.00 & 0.61 & 0.00 & 0.64 \\ \text{FileD} & 0.39 & 1.27 & 0.64 & 0.00 \end{array}$$

2) Unauthorized File Access Detector (UFAD): When access is denial, a record is logged in the access log. The UFAD is triggered by the event of this record being logged. Once triggered, The UFAD compares the username and filename from the record with the target usernames obtained from directory service and the target filenames obtained from the file management database. If there is a match, the UFAD extracts the timestamp, username, filename and access type

(R or W) from the record and saves this information as denial access information in the pool of denial access information. Subsequently, a request for access determination is made to the AACD.

3) Automatic Access Control Determinator (AACD): This component plays a crucial role in managing access privileges. When it is necessary to add access privileges, this component is responsible for actually adding these privileges to the ACL. Additionally, this component also has the role of removing privileges from the ACL when they are deemed unnecessary.

a) Addition of necessary access privileges: The AACD is activated upon receiving a determination request from the UFAD. Its role involves making access privileges determinations using information on access denial and the correlation between files. Specifically, it determines based on the correlation between files that a user has already the access privilege to and those that have been denied access. If the determination results in granting, the privileges are added on the ACL. In case of denial, the user is advised to request a manual determination if truly necessary. The AACD continues this process as long as there is data in the pool of denial access information.

The determination method is as follows. If the elements of the adjacency matrix exceed a certain threshold, it is considered correlated. The formula is shown in (3). Here, as an example, the threshold is set at 0.8 or higher.

$$\mathrm{Matrix}(\mathrm{File}_{\mathrm{old}}, \mathrm{File}_{\mathrm{new}}) \geq 0.8 \tag{3}$$

where $\mathrm{Matrix}(\mathrm{File}_{\mathrm{old}}, \mathrm{File}_{\mathrm{new}})$ is the correlation between files, $\mathrm{File}_{\mathrm{new}}$ is the new file to be accessed by user u, and $\mathrm{File}_{\mathrm{old}}$ is the file already accessed by user u.

b) Revocation of unnecessary access privileges: When the date changes, the AACC fetches records of access logs for a certain period. The AACD then fetches these records from the AACC. Additionally, the AACD fetches the current access privileges for each user from the capability database. The AACD compares the records with the current access privileges, identifying user names, access types (R or W), and file names that are present in the access privileges but not in the records. It indicates that the user has not accessed those files with those privileges for a certain period. The AACD then revocates the identified user names, access types, and file names from the ACL. Furthermore, the AACD also revocates these access privileges for the identified user names, access types, and file names from the capability database.

4) Manual Access Control Determinator (MACD): If the result of the automatic determination is denial, manual determination is carried out by the file owner in response to the user request. The MACD serves as the receiver of the access-privilege request from the user and the sender of the manual access determination request to the owner of the file. The user's request includes a Timestamp, Username, Filename, Access type (R or W), and the reason for the access privilege request. Upon receiving a request, the MACD retrieves the owner information of the specified Filename from the file server. Additionally, the file owner's contact information is obtained from the directory service. This ensures that the MACD can effectively communicate the manual access determination request to the file owner, equipped with all necessary details about the user's request and the file in question.

5) Directory Service: This component store user information. This setup assumes the use of pre-existing components within the system environment prior to the implementation of the proposed system. For example, AD can be considered as such a component. The information stored in this component includes user names, ranks, affiliations, and contact information.

6) File Management Database: This component stores the names of files targeted by the proposed system. The proposed system uses this component to decide whether a file is to be determineed.

7) Pool of Denial Access Information: This component stores information about denial access. This includes the timestamp, username, filename, and accesstype (R or W) all of which are in the access record. The UFAD is responsible for saving this information in this component. After a determination is made, the AACD then deletes this information from this component.

8) Graph Adjacency Matrix: This component stores the adjacency matrices of the graph. For each rank, matrices for each access type (R or W) are saved. These adjacency matrices are calculated by the AACC.

9) Capability: This component stores the capabilities for each user. These lists include the access privileges owned by each user, specifying the filenames and the accesstypes (R or W).

E. Issues on Scalability and Efficiency

After the proposed system was designed, the performance of the proposed system was tested in a brief environment. As a result, it turned out that there were issues regarding system scalability and efficiency [1]. Therefore, we modified the proposed system to enhance its efficiency and confirmed its improved scalability.

The first issue is the method of record detection. When a large number of denials are incurred, all their records are stored in an event log. The previous system maintained detected denial records as history. In each detection, records that the history did not have were fetched as missed detection. When many denial records are incurred in the event log file, it causes a heavy system load that the proposed system might not identify all missed detections, and the situation cannot be handled correctly. To address this, we refined the detection method to improve efficiency and ensure scalability, significantly reducing system load and improving detection accuracy.

The second issue is the method of record filtering. The proposed system could have caused delays in system processing. Record filtering is done by the target user name and file name. In addition to access logs, a large number of other logs are generated in the system environment. Thus, it is necessary to filter the target records among them. In the previous study, the records were filtered by a component responsible for

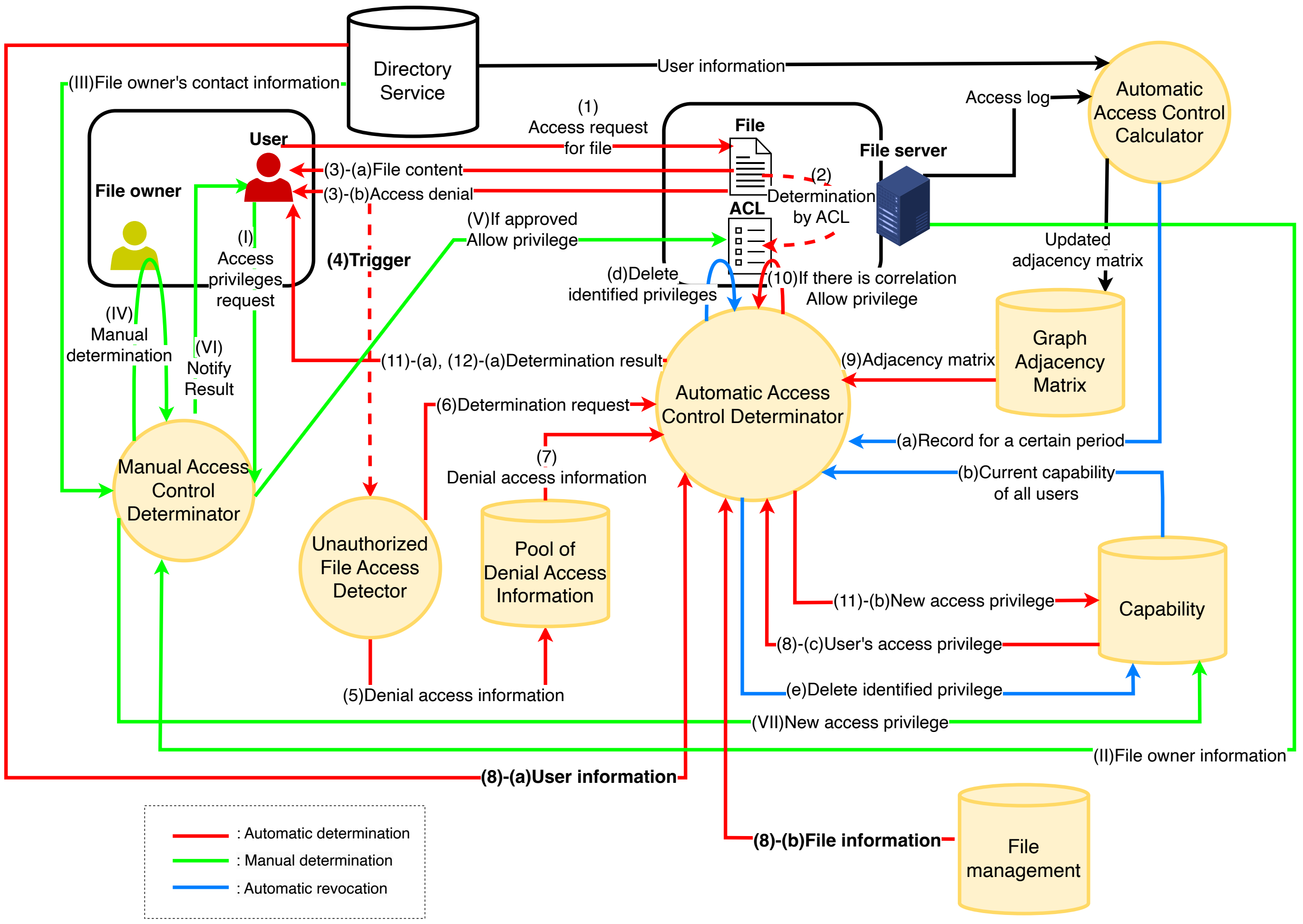
Directory Service
User information
(III)File owner's contact information
Automatic Access Control Calculator
Access log
(1) Access request for file
File
File server
User
File owner
(3)-(a)File content
(3)-(b)Access denial
(2) Determination by ACL
ACL
(I) Access privileges request
(4)Trigger
(V)If approved Allow privilege
(d)Delete identified privileges
(10)If there is correlation Allow privilege
Updated adjacency matrix
(IV) Manual determination
(VI) Notify Result
(11)-(a), (12)-(a)Determination result
(9)Adjacency matrix
Graph Adjacency Matrix
Automatic Access Control Determinator
(6)Determination request
(a)Record for a certain period
(7) Denial access information
(b)Current capability of all users
Manual Access Control Determinator
Unauthorized File Access Detector
Pool of Denial Access Information
(11)-(b)New access privilege
Capability
(8)-(c)User's access privilege
(5)Denial access information
(e)Delete identified privilege
(VII)New access privilege
(II)File owner information
(8)-(a)User information
(8)-(b)File information
File management
: Automatic determination
: Manual determination
: Automatic revocation

Fig. 5: Architecture of Modified System (yellow)

detection. If a large number of access denials occurred, the component was flooded with work. In the worst case, it could not be handled and could become fully dysfunctional. We have optimized the filtering process to handle high volumes more efficiently, preventing system delays and ensuring the component remains functional under increased load.

We modified the system's design regarding record management which traces the sequential number of the last record. The system thus retrieves the difference between the last and most recent records. Each record is assigned a sequential number by the operating system. By collecting not only the latest record but also the records representing these differences, the system prevents any missing denial records.

We also modified the system's design regarding record filtering for the detection component to specialize only in the detection of denial records. The detection component was detecting denial records and filtering whether they were target records or not. We gave that role to a component for determination. Thus, the load on the detection component is lightened.

However, this modification is expected to increase the load on the determination component. Since the above modification could increase the independence of the detection and determination components, When the load increases, a twist could be devised within each component. For example, in the detection component, if the number of detections is greater than the preset maximum number, only the maximum number of detections will be made and subsequent detections will be made after the process is completed. In the determination component, it is possible to record the determinations made during a certain period of time in the past and not determine the access types of the same user for the same file during that period of time. In this way, it could reduce the number of determinations.

These modifications have significantly improved the proposed system's efficiency and scalability, effectively addressing the initial concerns identified during the testing phase.

F. Architecture of Modified System

The architecture of modified system is shown in Fig. 5. The first change from the previous study is Step (4) in automatic determination of access privileges. After being triggered by denied events, the system collects attempts of denied access that have occurred since the last data collection, thereby

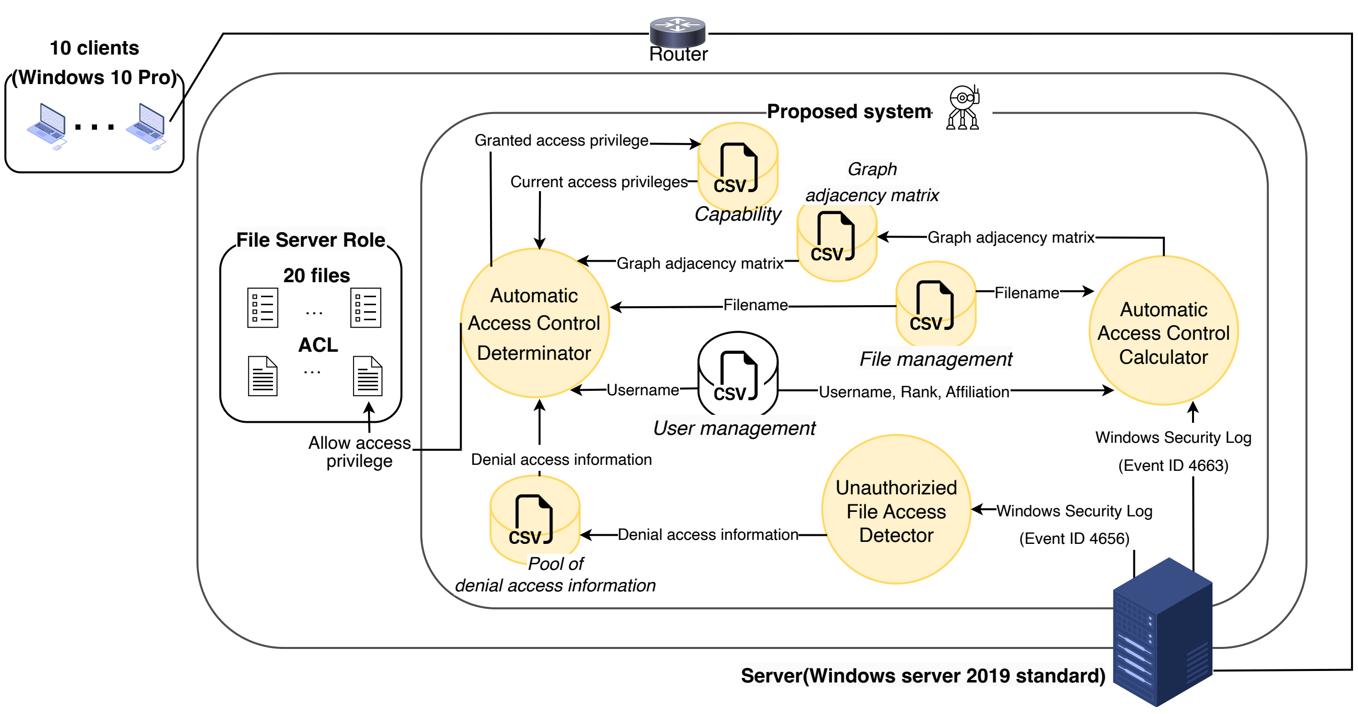
10 clients
(Windows 10 Pro)
Router
Proposed system
File Server Role
20 files
ACL
Allow access privilege
Granted access privilege
Current access privileges
Capability
Graph adjacency matrix
Graph adjacency matrix
Graph adjacency matrix
Automatic Access Control Determinator
Filename
Filename
File management
Automatic Access Control Calculator
Username
Username, Rank, Affiliation
User management
Windows Security Log (Event ID 4663)
Denial access information
Pool of denial access information
Denial access information
Unauthorizied File Access Detector
Windows Security Log (Event ID 4656)
Server(Windows server 2019 standard)
CSV

Fig. 6: Implementation of Proposed System (yellow)

ensuring that simultaneous access denials are also detected without omission. The second change is Step (8)-(a), (b) in automatic determination of access privileges. The role of log filtering is changed from the UFAD to the AACD, which lightens the workload of the UFAD and concentrates the tasks within the AACD, thereby ensuring scalability through efficient processing. The AACD executes record filtering on the basis of user and file information. If applicable, determinations are made; otherwise, the AACD deletes the access information without making a determination.

IV. Implementation of Proposed System

The proposed system was implemented in a more practical environment than previous work [1], because the previous environment was a simplified one, created on a single operating system on a single computer. We extended the system environment to address this limitation. The implemented environment is shown in Fig. 6. Ten machines were prepared to function as clients. Windows 10 Pro was installed as the operating system on each client machine. One user was assigned to each client machine. A single server was set up to serve both as a file server and to host the proposed system. Windows Server 2019 Standard was chosen as the operating system for the server. Twenty files, named from 00 to 19, were created on the file server. Each of the four DBs in the proposed system consists of CSV files.

The access log in Windows is "Security" in "Windows Log" (hereinafter referred to as Windows Security Log).

The following is a description of the setting for the system and the implementation of each component of the system.

A. Automatic Access Control Calculator (AACC)

This component has the role of graph calculation to represent the correlations among files and is implemented by Python in addition to the PowerShell command for fetching the Windows Security Log from the file server. It executes "Import-Csv" command to import Username from CSV File for User management and Filename from CSV File that stores target file information. It executes "Get-WinEvent" command to fetch the Windows Security Log from the file server. From the log, the "Where-Object" command retrieves 4663 event ID records of the target Username, Filename in the past certain period. The record with Event ID 4663 in the Windows Security Log indicates an attempt was made to access an object. Therefore, it extracts such records from the log. Then, it fetches information on Timestamp, Username, Filename, Accesstype (R or W) in the retrieved record. It executes "Sort-Object" command to sort the information based on Username and Timestamp.

For graph calculations, The "read_csv" function from the "pandas" library imports the user's Rank from CSV File for User management. Considering the rank, the sorted information is classified for each rank for each operation (R or W) determination graph. Using Algorithm 1, the graphs of each operation in each rank is calculated from the classified information. The "to_csv" function exports the graphs to the CSV Files for Graph adjacency matrix.

The graph calculation was set up in two stages. The first stage is calculated at 1:00 a.m. daily using data from 30 days prior to the previous day. The second stage is calculated every hour during business hours (9:00-17:00), using data up to the present time of the day. After the second stage of calculation, the graphs from the first and second stages were combined and normalized. This is because graph calculation takes a lot of time.

B. Unauthorized File Access Detector (UFAD)

In the file server, "Task Scheduler" monitors the Windows Security Log and checks records with Event ID 4656. The record with this Event ID means the request to an object, and it includes the result of the request. Task Scheduler executes the UFAD component implemented by Windows PowerShell script when the request failure record is detected.

The script consists of four PowerShell commands. Firstly, it executes "Get-WinEvent" command similar to the AACC. From the log, the "Where-Object" command searches the request failure record. Timestamp, Username, Filename, and Access type (R or W) are extracted from the failure record by the "Select-Object" command. Finally, "Export-Csv" command saves extracted data to a CSV file that serves as Pool of Denial Access Information DB.

C. Automatic Access Control Determinator (AACD)

This component performs automatic access determination using denied access information and is implemented in Python. The "read_csv" function similar to the AACC imports Username and Filename from the oldest denied access information stored in the CSV file for pool of denied access information. The function imports Username from CSV File for User management and Filename from CSV File that stores target file information. If Usename and Filename of the denied access information match with the target Username and Filename, the code fetch Username, Filename, Accesstype from the oldest denied access information. If they do not match, delete them from CSV file for pool of denied access information without determination.

The "read_csv" function imports the user's Rank from CSV File for User management. The function imports the user's access privileges from CSV File for Capability. The function imports the rank's adjacency matrix from CSV File for graph adjacency matrix. The code determines if the maximum value in the adjacency matrix exceeds the threshold value, where the row is the filename stored in the CSV File for Capability and the column is the Filename of the denied access information.

If the result of the determination is to allow, in the Python code, the "os.system ()" function is used to execute the Windows "icacls" command, which allows the user the access privilege for the file. The "socket" function from the "socket" library create a new socket object. The "connect" method of the socket object s connects to the specified IP address (HOST) and port number (PORT). The "send" method transmits a message to the connected server. This message includes Timestamp, Username,Filename and Accesstype (R or W). The "csv.writer" function from the "csv" library write Username, Filename and Accesstype (R or W) into CSV file for Capability as new access privilege.

If the result of the determination is to deny, a message is sent using the 'socket' function as in the case of allowed privileges. This message includes Timestamp, Username, Filename and Accesstype (R or W). Additionally, the user has the option to submit a new access privilege request along with the reason for the request.

Regardless of the result of either determination, the oldest denied access information is deleted. As long as data exists in the CSV file for pool of denied access information, the determination is repeated.

On each user's client side, Python code utilizing the "socket" function from the "socket" library is constantly running to receive the determination results. The "listen" method of the socket object s place the socket in server mode and wait for incoming connections. If data is received, the content of the message is displayed in the GUI using the "messagebox" from the "tkinter" library. Each client user is distinguished by IP address, and port number 8000 is reconfigured to open for this implementation.

D. CSV File for User Management

This file stored the user information in the following three columns.

- Username
- Rank
- Affiliation

E. CSV File that stores Target File Information

This file stored target file information in the following one column.

- Filename

F. CSV File for Pool of Denied Access Information

This file stored the denied log information in the following four columns.

- Timestamp
- Username
- Filename
- Accesstype (R or W)

G. CSV Files for Graph Adjacency Matrix

These files stored the adjacency matrices of the graph calculated by AACC. There exist two types of adjacency matrices for each rank, corresponding to two access types (R or W).

H. CSV File for Capability

This file stores information about the accesstype (R, R/W) for which the user has privileges to access the file.

- Username
- Filename
- Accesstype (R or W)

TABLE III: PRELIMINARY EXPERIMENTAL RESULTS: RESPONSE TIME

	Response Time (s)						
Number of Users	1st Trial	2nd Trial	3rd Trial	4th Trial	5th Trial	5-Trial Average	Average per User
Original System							
1(A)	6.85	6.22	6.62	5.96	6.69	6.47	6.47
2(A/B)	7.96	7.92	6.51	6.71	7.82	7.38	3.69
3(A/B/C)	7.91	9.56	7.34	9.49	8.58	8.58	2.86
4(A/B/C/D)	8.61	8.26	8.35	8.79	8.16	8.43	2.11
5(A/B/C/D/E)	8.79	8.95	9.56	9.10	9.46	9.17	1.83
6(A/B/C/D/E/F)	11.20	12.77	9.91	12.54	11.84	11.65	1.94
Modified System							
1(A)	6.46	5.19	5.16	6.58	4.94	5.58	5.58
2(A/B)	5.46	5.50	5.64	4.76	4.83	5.24	2.61
3(A/B/C)	6.04	5.32	5.68	5.24	8.95	6.25	2.08
4(A/B/C/D)	10.41	5.22	5.62	5.39	5.82	6.49	1.62
5(A/B/C/D/E)	6.17	10.61	5.83	6.19	5.15	6.79	1.36
6(A/B/C/D/E/F)	6.00	10.62	6.11	5.78	6.44	6.99	1.16

V. EVALUATION EXPERIMENT

The main objective of this paper was to verify the feasibility of the modified system in a real-world setting. The initial method of verification, which used a limited number of users and accesses, was found to be inadequate for a comprehensive evaluation. To address this, we implemented the improved system in a more practical environment and subjected it to high stress for feasibility testing. This was done to demonstrate that the proposed system, even when integrating processes from existing file systems, does not lose its feasibility.

A. *Preliminary Experiment*

We conducted preliminary experiments to compare the performance of both versions of our system (original and modified). Performance was compared by conducting the verification under the same conditions as those used in our previous study [1].

1) System Environment: Both version of the system were implemented in a experimental environment. We used Intel® NUC 8 Pro Kit (NUC8v7PNH) as the hardware and Windows 11 Pro as the operating system (OS). We set up six users (A$\cdots$F) on the OS and ten files to be accessed by the user.

2) Methodology: We measured the determination time of the both systems as a performance test. We measured the response time from when users access unauthorized files to when determination results are notified to the users.

3) Results: The results of the verification experiments are listed in Table III. Number of Users columns indicates the number and name of accessing users. The columns 1st Trial through 5th trial show the response times for each number of users. If the number of users is 2 or more, the latest response time among users is noted. The 5-trial Average column shows the average of five trials for the same number of users. The average per user is calculated by dividing the five-trial average by the number of users in the corresponding row.

The results indicate improved response times for both 5-trial average and average per user. Despite the preliminary experiment not supposing high-load conditions, the improvement in the response time was observed.

B. *Experimental Environment*

We built a more practical environment to verify the scalability of the modified system. The system environment is shown in Fig. 7. Ten machines were prepared as clients. Each client had Windows 10 as its operating system and segmented on the network into subgroups. Each client machine was assigned to one user, with each user being named from user_A to user_J. One server was also prepared with the active directory, file server, and the proposed system. Twenty files with file names 00 through 19 were created on the file server.

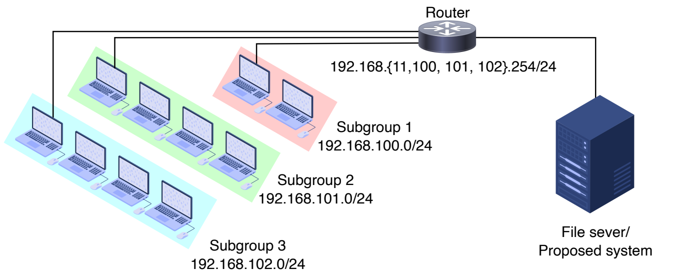

Fig. 7: Experimental Environment

Table IV shows the privileges allowed to each user for each file. Each user was assigned a job rank. As noted next to the user name in the user column of Table IV, we assigned rank 2 to users A, D, and H, and assigned rank 1 to the rest. It is assumed that rank 2 is higher than rank 1. Each user's rank is taken into account when calculating the graph.

C. *Dataset*

Graph calculation requires data on each user's access history. A simulated data set was generated for this experiment. The following steps were taken to generate the simulated data set.

1) Set the column items of the simulated data set to be generated
   - Timestamp
   - Username
   - Filename

TABLE IV: ACCESS PRIVILEGES FOR EACH USER IN A PRACTICAL ENVIRONMENT

UserName	**File_00**	**File_01**	**File_02**	**File_03**	**File_04**	**File_05**	**File_06**	**File_07**	**File_08**	**File_09**
User_A (2)			R	R	R		R	R/W	R/W	R/W
User_B (1)			R/W	R/W	R/W		R/W			R/W
User_C (1)			R/W	R/W	R/W		R/W			R/W
User_D (2)		R		R		R	R	R	R/W	
User_E (1)		R/W		R/W		R/W	R/W			
User_F (1)		R/W		R/W		R/W	R/W			
User_G (1)		R/W		R/W		R/W	R			
User_H (2)	R				R	R	R/W	R/W	R/W	R
User_I (1)	R/W				R/W	R/W	R/W			R
User_J (1)	R/W				R/W	R/W	R/W			R

UserName	**File_10**	**File_11**	**File_12**	**File_13**	**File_14**	**File_15**	**File_16**	**File_17**	**File_18**	**File_19**
User_A (2)	R			R			R			R
User_B (1)	R			R/W			R/W	R		R/W
User_C (1)	R		R/W			R/W		R	R/W	
User_D (2)	R/W		R/W		R	R		R	R	
User_E (1)	R/W				R/W	R/W				R/W
User_F (1)	R/W		R		R				R/W	
User_G (1)	R/W					R	R/W	R/W		
User_H (2)		R/W	R	R/W		R	R	R/W	R	
User_I (1)		R/W	R		R		R		R	R
User_J (1)		R/W		R	R/W		R	R	R/W	

TABLE V: DATA SET SUMMARY FOR GRAPH CALCULATION (read)

User	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
A(2)	0	0	17	18	21	0	17	38	35	36	22	0	0	22	0	0	18	0	0	18
B(1)	0	0	19	27	32	0	35	0	0	14	20	0	0	27	0	0	29	17	0	30
C(1)	0	0	20	30	25	0	33	0	0	28	13	0	31	0	0	27	0	10	26	0
D(2)	0	26	0	17	0	35	21	21	30	0	37	0	31	0	29	15	0	22	16	0
E(1)	0	25	0	27	0	24	31	0	0	0	32	0	0	0	33	23	0	0	0	37
F(1)	0	28	0	56	0	37	34	0	0	0	36	0	19	0	23	0	0	0	29	0
G(1)	0	33	0	45	0	26	16	0	0	0	34	0	0	0	0	23	33	40	0	0
H(2)	15	0	0	0	18	10	29	20	24	14	0	24	19	20	0	11	15	23	15	0
I(1)	26	0	0	0	34	32	25	0	0	15	0	22	25	0	20	0	20	0	11	20
J(1)	33	0	0	0	24	24	27	0	0	21	0	19	0	13	27	0	14	19	23	0

TABLE VI: DATA SET SUMMARY FOR GRAPH CALCULATION (write)

User	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
A(2)	0	0	0	0	0	0	0	17	14	13	0	0	0	0	0	0	0	0	0	0
B(1)	0	0	7	11	12	0	13	0	0	5	0	0	0	16	0	0	14	0	0	12
C(1)	0	0	9	16	13	0	11	0	0	11	0	0	14	0	0	13	0	0	12	0
D(2)	0	0	0	0	0	0	0	0	15	0	19	0	12	0	0	0	0	0	0	0
E(1)	0	10	0	10	0	7	12	0	0	0	14	0	0	0	12	5	0	0	0	14
F(1)	0	12	0	22	0	21	12	0	0	0	20	0	0	0	0	0	0	0	14	0
G(1)	0	11	0	24	0	12	0	0	0	0	13	0	0	0	0	0	11	18	0	0
H(2)	0	0	0	0	0	0	12	5	13	0	0	11	0	7	0	0	0	11	0	0
I(1)	12	0	0	0	15	19	3	0	0	0	0	13	0	0	0	0	0	0	0	0
J(1)	7	0	0	0	12	7	6	0	0	0	0	8	0	0	9	0	0	0	10	0

- Accesstype (R or W)

2) Set the date and time of the simulated data set to be generated. In this study, past month (past 30 days).
3) Generate simulated log data set
   a) Set the time as 9:00–17:00
   b) Set the probability of generating a log every 15 min
      - On the hour: 70%
      - 15 min past the hour: 50%
      - 30 min past the hour: 30%
      - 45 min past the hour: 20%

      If not generated, move to the next hour (e.g., if not generate at 9:15 $\rightarrow$ 10:00)
   c) Select file
      i) Set weight to each file
         - File the user does not have privileges to: 0
         - File the user has read privileges to: 0.3
         - File the user has read/write privileges to: 0.5
      ii) Calculate sum of these weights per user
      iii) Generate a random number from 0 to total sum
      iv) Select the file in which the accumulated weight exceeds the random number for the first time
   d) Select access type

      If read is generated and the user has read/write privileges for the file, there is a 40% chance of generating write in the next 15 min

      If not, generate read at a probability in (b)
   e) Repeat the above steps for each user

TABLE VII: ADJACENCY MATRIX (rank1_read)

	00	01	02	03	04	05	06	09	10	11	12	13	14	15	16	17	18	19
00	0.00	0.00	0.00	0.00	0.10	0.09	0.14	0.19	0.00	0.18	0.13	0.15	0.14	0.00	0.22	0.14	0.10	0.11
01	0.00	0.00	0.00	0.18	0.00	0.27	0.20	0.00	0.27	0.00	0.00	0.00	0.13	0.12	0.08	0.17	0.15	0.06
02	0.00	0.00	0.00	0.17	0.18	0.00	0.12	0.09	0.01	0.00	0.20	0.20	0.00	0.11	0.08	0.09	0.13	0.13
03	0.00	0.18	0.17	0.00	0.08	0.18	0.30	0.08	0.23	0.00	0.19	0.14	0.25	0.22	0.21	0.14	0.14	0.17
04	0.10	0.00	0.18	0.08	0.00	0.10	0.21	0.11	0.26	0.03	0.17	0.13	0.16	0.10	0.10	0.21	0.21	0.14
05	0.09	0.27	0.00	0.18	0.10	0.00	0.26	0.16	0.10	0.27	0.13	0.00	0.17	0.11	0.23	0.07	0.19	0.16
06	0.14	0.20	0.12	0.30	0.21	0.26	0.00	0.23	0.25	0.12	0.27	0.12	0.16	0.11	0.18	0.30	0.14	0.14
09	0.19	0.00	0.09	0.08	0.11	0.16	0.23	0.00	0.05	0.19	0.10	0.04	0.00	0.07	0.11	0.09	0.18	0.14
10	0.00	0.27	0.01	0.23	0.26	0.10	0.25	0.05	0.00	0.00	0.04	0.05	0.19	0.15	0.15	0.10	0.06	0.18
11	0.18	0.00	0.00	0.00	0.03	0.27	0.12	0.19	0.00	0.00	0.00	0.00	0.14	0.00	0.10	0.15	0.17	0.02
12	0.13	0.00	0.20	0.19	0.17	0.13	0.27	0.10	0.04	0.00	0.00	0.00	0.10	0.14	0.03	0.07	0.15	0.09
13	0.15	0.00	0.20	0.14	0.13	0.00	0.12	0.04	0.05	0.00	0.00	0.00	0.15	0.00	0.11	0.01	0.10	0.18
14	0.14	0.13	0.00	0.25	0.16	0.17	0.16	0.00	0.19	0.14	0.10	0.15	0.00	0.23	0.03	0.02	0.15	0.20
15	0.00	0.12	0.11	0.22	0.10	0.11	0.11	0.07	0.15	0.00	0.14	0.00	0.23	0.00	0.05	0.07	0.05	0.13
16	0.22	0.08	0.08	0.21	0.10	0.23	0.18	0.11	0.15	0.10	0.03	0.11	0.03	0.05	0.00	0.22	0.09	0.09
17	0.14	0.17	0.09	0.14	0.21	0.07	0.30	0.09	0.10	0.15	0.07	0.01	0.02	0.07	0.22	0.00	0.02	0.05
18	0.10	0.15	0.13	0.14	0.21	0.19	0.14	0.18	0.06	0.17	0.15	0.10	0.15	0.05	0.09	0.02	0.00	0.00
19	0.11	0.06	0.13	0.17	0.14	0.16	0.14	0.14	0.18	0.02	0.09	0.18	0.20	0.13	0.09	0.05	0.00	0.00

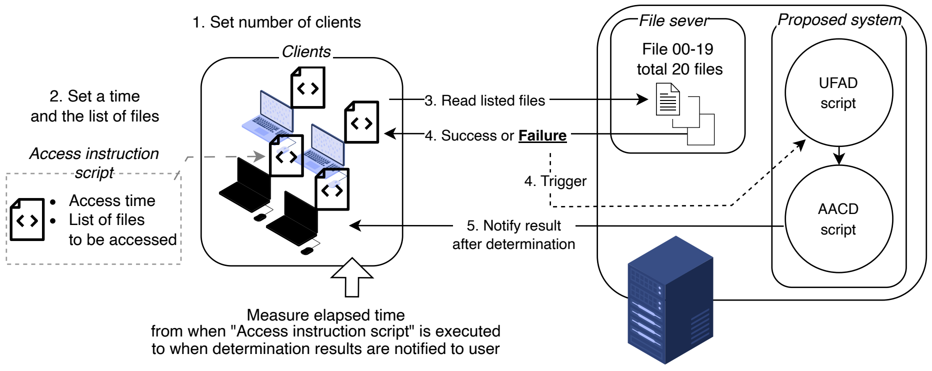

Fig. 8: Experimental Methodology

A summary of the dataset exists. The summary concerning read is shown in Table V, while the summary about write is shown in Table VI.

Graphs for access determination are calculated from the generated access logs. These graphs are shown across multiple tables. Graphs related to rank 1 are shown in Table VII for read determination and in Table VIII for write determination. Similarly, graphs related to rank 2 are displayed in Table IX for read determination and in Table X for write determination. The method for creating graphs is described in the function of the AACC. The graphs for read determination are calculated from the access histories of the same or lower rank users. The higher rank users tend to have more files represented on the graph compared to the lower rank users. The graphs for write determination are calculated from the access histories of the same rank users. As the number of the same rank users decreases for higher ranks, there is a higher likelihood of having fewer files represented on the graph.

D. Methodology

The experimental methodology is shown in Fig. 8. Each client has an "Access instruction script" for accessing files in the file server. We can set a time of access and a list of accessing files to the script. The verification experiment was conducted using the following steps.

1) Set the number of clients
(Choosing from 1 to 10 clients)
2) Set the time and files to be accessed
(Selecting 1 to 20 files) to the scripts of selected clients
3) Clients read all files at once when time is up by the script
4) The File server detects denial logs of accesses and executes the UFAD
5) The AACD makes determinations based on the logs and notifies the user

At each client, we measured the elapsed time as a response time from the Access instruction script starts to it receives the notification.

Total Access is the number of accessing users multiplied by the number of files listed. For example, a total of 100 accesses is 10 users accessing 10 files from 00 to 09. The number of accesses was increased from 1 to 10 in increments of 1. After 10, the number of accesses was increased in increments of 10 up to 200.

Accesses were done sequentially to the listed files. Although not strictly simultaneous accesses, the next files were accessed with 0.2-s delay. This condition simulated an intense load on our modified system. When the maximum number of access

TABLE VIII: ADJACENCY MATRIX (rank1_write)

	00	01	02	03	04	05	06	09	10	11	12	13	14	15	16	17	18	19
00	0.00	0.00	0.00	0.00	0.19	0.33	0.20	0.00	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.00
01	0.00	0.00	0.00	0.14	0.00	0.25	0.26	0.00	0.46	0.00	0.00	0.00	0.11	0.00	0.00	0.43	0.21	0.29
02	0.00	0.00	0.00	0.04	0.18	0.00	0.28	0.01	0.00	0.00	0.21	0.36	0.00	0.09	0.17	0.00	0.23	0.37
03	0.00	0.14	0.04	0.00	0.05	0.45	0.36	0.41	0.30	0.00	0.34	0.43	0.10	0.19	0.54	0.29	0.34	0.09
04	0.19	0.00	0.18	0.05	0.00	0.29	0.09	0.21	0.00	0.16	0.17	0.25	0.02	0.06	0.19	0.00	0.36	0.00
05	0.33	0.25	0.00	0.45	0.29	0.00	0.32	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.30	0.21	0.06	0.23
06	0.20	0.26	0.28	0.36	0.09	0.32	0.00	0.15	0.28	0.19	0.13	0.10	0.22	0.00	0.09	0.00	0.23	0.28
09	0.00	0.00	0.01	0.41	0.21	0.00	0.15	0.00	0.00	0.00	0.01	0.00	0.00	0.29	0.04	0.00	0.19	0.00
10	0.00	0.46	0.00	0.30	0.00	0.00	0.28	0.00	0.00	0.00	0.00	0.00	0.00	0.48	0.08	0.22	0.04	0.01
11	0.39	0.00	0.00	0.00	0.16	0.56	0.19	0.00	0.00	0.00	0.00	0.00	0.41	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.21	0.34	0.17	0.00	0.13	0.01	0.00	0.00	0.00	0.00	0.00	0.41	0.00	0.00	0.22	0.00
13	0.00	0.00	0.36	0.43	0.25	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00
14	0.00	0.11	0.00	0.10	0.02	0.00	0.22	0.00	0.00	0.41	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44
15	0.00	0.00	0.09	0.19	0.06	0.00	0.00	0.29	0.48	0.00	0.41	0.00	0.00	0.00	0.00	0.00	0.10	0.28
16	0.00	0.00	0.17	0.54	0.19	0.30	0.09	0.04	0.08	0.00	0.00	0.14	0.00	0.00	0.00	0.17	0.00	0.00
17	0.00	0.43	0.00	0.29	0.00	0.21	0.00	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.00	0.00
18	0.23	0.21	0.23	0.34	0.36	0.06	0.23	0.19	0.04	0.00	0.22	0.00	0.00	0.10	0.00	0.00	0.00	0.00
19	0.00	0.29	0.37	0.09	0.00	0.23	0.28	0.00	0.01	0.00	0.00	0.00	0.44	0.28	0.00	0.00	0.00	0.00

TABLE IX: ADJACENCY MATRIX (rank2_read)

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
00	0.00	0.00	0.00	0.00	0.13	0.07	0.16	0.03	0.08	0.14	0.00	0.14	0.11	0.10	0.11	0.02	0.19	0.11	0.09	0.09
01	0.00	0.00	0.00	0.19	0.00	0.21	0.16	0.07	0.10	0.00	0.24	0.00	0.03	0.00	0.10	0.13	0.06	0.16	0.17	0.05
02	0.00	0.00	0.00	0.16	0.18	0.00	0.09	0.04	0.15	0.12	0.04	0.00	0.13	0.16	0.00	0.07	0.09	0.06	0.09	0.09
03	0.00	0.19	0.16	0.00	0.06	0.14	0.26	0.07	0.06	0.11	0.20	0.00	0.17	0.07	0.21	0.16	0.20	0.13	0.13	0.17
04	0.13	0.00	0.18	0.06	0.00	0.08	0.22	0.10	0.06	0.11	0.24	0.09	0.12	0.15	0.12	0.07	0.13	0.17	0.17	0.19
05	0.07	0.21	0.00	0.14	0.08	0.00	0.21	0.08	0.06	0.12	0.16	0.19	0.15	0.01	0.25	0.13	0.18	0.09	0.20	0.12
06	0.16	0.16	0.09	0.26	0.22	0.21	0.00	0.14	0.15	0.21	0.22	0.15	0.22	0.14	0.13	0.14	0.15	0.26	0.11	0.12
07	0.03	0.07	0.04	0.07	0.10	0.08	0.14	0.00	0.17	0.17	0.06	0.06	0.09	0.12	0.08	0.04	0.08	0.03	0.09	0.03
08	0.08	0.10	0.15	0.06	0.06	0.06	0.15	0.17	0.00	0.09	0.16	0.07	0.10	0.16	0.05	0.05	0.08	0.06	0.08	0.05
09	0.14	0.00	0.12	0.11	0.11	0.12	0.21	0.17	0.09	0.00	0.06	0.14	0.07	0.12	0.00	0.05	0.13	0.08	0.12	0.16
10	0.00	0.24	0.04	0.20	0.24	0.16	0.22	0.06	0.16	0.06	0.00	0.00	0.10	0.06	0.21	0.14	0.13	0.17	0.07	0.16
11	0.14	0.00	0.00	0.00	0.09	0.19	0.15	0.06	0.07	0.14	0.00	0.00	0.05	0.05	0.09	0.04	0.15	0.12	0.14	0.01
12	0.11	0.03	0.13	0.17	0.12	0.15	0.22	0.09	0.10	0.07	0.10	0.05	0.00	0.02	0.13	0.17	0.02	0.09	0.19	0.06
13	0.10	0.00	0.16	0.07	0.15	0.01	0.14	0.12	0.16	0.12	0.06	0.05	0.02	0.00	0.08	0.06	0.11	0.05	0.11	0.12
14	0.11	0.10	0.00	0.21	0.12	0.25	0.13	0.08	0.05	0.00	0.21	0.09	0.13	0.08	0.00	0.17	0.02	0.02	0.11	0.17
15	0.02	0.13	0.07	0.16	0.07	0.13	0.14	0.04	0.05	0.05	0.14	0.04	0.17	0.06	0.17	0.00	0.03	0.13	0.05	0.09
16	0.19	0.06	0.09	0.20	0.13	0.18	0.15	0.08	0.08	0.13	0.13	0.15	0.02	0.11	0.02	0.03	0.00	0.21	0.08	0.09
17	0.11	0.16	0.06	0.13	0.17	0.09	0.26	0.03	0.06	0.08	0.17	0.12	0.09	0.05	0.02	0.13	0.21	0.00	0.05	0.04
18	0.09	0.17	0.09	0.13	0.17	0.20	0.11	0.09	0.08	0.12	0.07	0.14	0.19	0.11	0.11	0.05	0.08	0.05	0.00	0.00
19	0.09	0.05	0.09	0.17	0.19	0.12	0.12	0.03	0.05	0.16	0.16	0.01	0.06	0.12	0.17	0.09	0.09	0.04	0.00	0.00

TABLE X: ADJACENCY MATRIX (rank2_write)

	06	07	08	09	10	11	12	13	17
06	0.00	0.37	0.00	0.00	0.00	0.00	0.00	0.00	0.83
07	0.37	0.00	0.65	1.17	0.00	0.45	0.00	0.00	0.08
08	0.00	0.65	0.00	0.36	0.41	0.44	0.36	0.57	0.37
09	0.00	1.17	0.36	0.00	0.00	0.00	0.00	0.00	0.00
10	0.00	0.00	0.41	0.00	0.00	0.00	1.46	0.00	0.00
11	0.00	0.45	0.44	0.00	0.00	0.00	0.00	0.00	0.66
12	0.00	0.00	0.36	0.00	1.46	0.00	0.00	0.00	0.00
13	0.00	0.00	0.57	0.00	0.00	0.00	0.00	0.00	0.82
17	0.83	0.08	0.37	0.00	0.00	0.66	0.00	0.82	0.00

instances, which is 200, was reached, 200 accesses occurred within about 4 seconds.

The determination was made using graphs that were already calculated. In fact, the graphs could be recalculated at the same time as the determination. However, to align the experimental conditions, the graphs were generated under fixed conditions. Therefore, the determination time is not affected by the load due to recalculation.

E. Result

The results are listed in Table XI. Total Access is the total number of files listed in each user's script file. Unauthorized Access (UA) is access to files for which a user does not have read privileges among all accesses. Detected UAs are accesses detected with our system among UAs. Missed detection is the number of cases that our system failed to detect even though access logs were generated. Double detection is the number of cases in which our system detected double access logs for a single access log. The first response time refers to the response time of the proposed system for the first UA. Similarly, the

TABLE XI: EXPERIMENTAL RESULTS

Total Access (cases)	Unauthorized Access (cases)	Detected UA (cases)	Missed Detection (cases)	Double Detection (cases)	First Response Time (s)	Last Response Time (s)	Average Response Time (s)
1	1	1	0	0	2.97	2.97	2.97
10	7	7	0	0	3.22	3.27	3.24
50	27	27	0	0	4.05	5.14	4.50
100	48	48	0	0	5.03	8.32	7.06
150	74	74	0.67	0.67	4.73	10.24	8.87
200	97	97.33	0.33	0.67	5.97	13.56	12.05

last response time refers to the response time of the proposed system for the last UA. The average response time is calculated as the mean of the response times for each determination made with the system. The same trial was conducted three times for each total accesses. Each value in Table XI represents the average of three trials. Therefore, some values are decimal.

F. Discussion

From the results of our verification, we highlight three key points regarding our modified system. The first point concerns its scalability. The total number of accesses was varied from 1 to 200. The verification was conducted under conditions that placed a heavy burden on the system, with approximately half of the accesses being unauthorized. As a result, we achieved a detection rate of over 98% for up to 200 cases, despite some cases of missed or double detections.

The second point concerns the efficiency of the system. The average response time was 2.97s with a single access. In contrast, when the total number of accesses reached 200, the average response time increased to 12.05s. Even though the number of accesses went up by 200 times, the processing time only increased 4 times. This suggests that our system operates efficiently under heavy-load conditions.

The third point concerns missed and double detection. Missed and double detection began to occur after the total number of accesses exceeded 80. Although those numbers are less than 2%, they need to be corrected to improve the detection accuracy.

1) Limitation: We verified the efficiency and scalability of our modified access-control system, but not its effectiveness. It is necessary to show that our system could allow necessary access and deny unnecessary access. Additionally, the proposed method is considered to work well in team-oriented tasks, but its effectiveness and the optimal team size for its application have not been proven.

The modified system infers correlation among files only from user access patterns. There is still potential to investigate whether additional factors could be incorporated to infer more accurate correlations.

VI. Conclusion and Future Work

We implemented a modified version of our previously proposed access-control system in a practical environment to verify its scalability by applying a high load to it. This is because the previous implementation and verification were done in a simplified environment.

The verification results indicate that the system works under high loads. Comparing the rate of increase in the number of accesses to that in response time, the low rate of increase in response time indicates that the system is capable of efficient processing.

The evaluation results indicate that there are still some detection errors and double detections, so that the system needs further improvement to increase detection accuracy. The feasibility of this system has not yet been validated, and there is still potential for improvement to infer correlations. To address these issues, we will evaluate the effectiveness of the proposed method in teams of different sizes. Therefore, we plan to further improve this system from this perspective.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Numbers JP19K20268, JP24K14959.

References

[1] Y. Kodaka, H. Hasegawa, and H. Takakura, “Design and implementation of access control method based on correlation among files,” in Proceedings of The 16th International Conference on Advances in Human oriented and Personalized Mechanisms, Technologies, and Services, pp. 44-51, 2023.

[2] P. Samarati and S. C. Vimercati, “Access control: policies, models, and mechanisms,” Foundations of Security Analysis and Design, R. Focardi, R. Gorrieri, ed., Springer, pp. 137-196, 2001.

[3] D. F. Ferraiolo and D. R. Kuhn, “Role-based access control,” 15th National Computer Security Conference, pp. 554-563, 1992.

[4] V. C. Hu et al., “Guide to Attribute Based Access Control (ABAC) definition and considerations,” U.S. Department of Commerce, 2014.

[5] H. Xia, M. Dawande, and V. Mookerjee, “Role refinement in access control: model and analysis,” INFORMS Journal on Computing vol. 26, no. 4, pp. 866-884, 2014.

[6] L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea, “Real life challenges in access-control management,” in Proceedings of the CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, pp. 899-908, 2009.

[7] D. E. Bell and L. J. LaPadula, “Secure computer systems: mathematical foundation report ESD-TR-73-275,” MITRE Corp., 1973.

[8] Ponemon Institute, “2022 cost of insider threats global report,” Proofpoint, 2022.

[9] D. Tsiostas, and N. Chouliaras, and I. Kantzavelou, and L. Maglaras, and C. Douligeris, and V. Vlachos, “The insider threat: reasons, effects and mitigation techniques,” in 24th Pan-Hellennic Conference on Informatics, pp. 340-345, 2020.

[10] Y. Kodaka, H. Hasegawa, and H. Takakura, “A proposal for access control method based on file relation inference from users behavior (in Japanese),” IEICE Technical Report vol. 123, no. 86, pp. 40-47, 2023.

[11] T. Xu, H. M. Naing, L. Le and Y. Zho, “How do system system administrators resolve access-denied issues in the real world?,” in Proceedings of the CHI Conference on Human Factors in Computing Systems, pp. 348-361, 2017.

[12] M. Beckerle and L. A. Martucci, “Formal definitions for usable access control rule sets from goals to metrics,” in Proceedings of the Ninth Symposium on Usable Privacy and Security, pp. 1-11, 2013.
[13] M. Mazurek, and P. Klemperer, and R. Shay, and H. Takabi, and L. Bauer, and L.Cranor, “Exploring reactive access control,” in Proceedings of the CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, pp. 2085-2094, 2011.
[14] N. Shalev, I. Keidar, Y. Weinsberg. Y. Moatti, and E. B. Yehuda, “WatchIT: who watches your IT guy,” in Proceedings 26th Symposium on Operating Systems Principles, pp. 515-530, 2017.
[15] Y. Desmedt and A. Shaghaghi, “Function-based access control (FBAC) from access control matrix to access control tensor,” From Database to Cyber Security, vol 11170, pp. 143-165, 2018.
[16] H. Mannila, H. Toivonen, and A. Verkamo, “Discovery of frequent episodes in event sequence,” Data Mining and Knowledge Discovery, vol. 1, pp. 259-289, 1997.
[17] B. Camina, R. Monroy, L. Trejo, and E. Sanchez, “Towards building a masquerade detection method based on user file system navigation,” in Proceedings of the 10th Mexican International Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol. 7094, pp. 174-186, 2011.
[18] B. Camina, J. Rodriguez, and R. Monroy, “Towards a masquerade detection system based on user’s tasks,” in Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses, Lecture Notes in Computer Science, vol. 8688, pp. 447-465, 2014.
[19] S. Huang, Z. Cao, C. Raines, M. Yang, and C. Simon, “Detecting intruders by user fie access patterns,” in Proceedings of the 13th International Conference on Network and System Security, Lecture Notes in Computer Science, vol. 11928, pp. 320-335, 2019.
[20] S. Mehnaz and E. Bertino, “A fine grained approach for anomaly detection in file system accesses with enhanced temporal user profiles,” IEEE Transactions on Dependable and Secure Computing, vol. 18, Issue 6, pp. 2535-2550, 2021.
[21] Y. Chen and B. Malin, “Detection of anomalous insiders in collaborative environments wia relational analysis of access logs,” in Proceedings of the 1st ACM conference on Data and application security and privacy, pp. 63-74, 2011.