IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

AI-BASED FILE ACCESS PREDICTION SYSTEM FOR SECURE INFORMATION STORAGE

Adithya Pratheep[1], Vidya Sree DV[2], Dr. P. Durgadevi[3]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[1,2]

Mentor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[3]

Abstract: Every time someone opens, edits or downloads a file, a record is written somewhere. Over weeks and months those records pile up far beyond what an administrator can read line by line, and odd behaviour is easy to miss in all that noise. This paper describes a prototype that looks at the context around each access event and labels it as normal or suspicious. The inputs include the time of access, IP address, location, how often the user requests files, how sensitive the file is, and what the user has done before. Once the records have been checked and cleaned, they go to a Random Forest classifier. A risk and alert layer acts on its output, and an audit store keeps every event with its prediction so it can be reviewed later. A Streamlit dashboard brings activity, predictions, risk levels and alerts together in one place. The system sits beside existing authentication and authorization controls and does not try to replace them. We present it as a working prototype and a methodology, not as a finished product. We report no accuracy figures, because the model has not yet been tested on an independently prepared dataset. The sections that follow cover the motivation, related work, architecture, feature preparation, prediction workflow, testing plan and directions for future work.

Keywords: file-access prediction, information storage security, Random Forest, anomaly detection, user behaviour analysis, audit logging.

1. INTRODUCTION

Information Storage Management is about how digital information is kept, organised, protected and retrieved, and these days almost every organisation leans on shared drives, databases and cloud storage to do it. As the number of files grows, so does the number of requests to reach them, and each request leaves a log entry behind. The volume is more than an administrator can realistically inspect by hand, which means a genuine security incident can sit unnoticed among thousands of routine operations. Authentication and authorization remain the foundation of storage security, since they establish who is asking and what that person is allowed to use [1]. Even so, they leave a gap. A legitimate account can still behave strangely: someone might open a sensitive file in the middle of the night, sign in from a place they have never connected from, or pull an unusually large number of files in a short time, and a simple permission check says nothing about any of it. Reading logs manually to catch such cases takes a long time, and some patterns only show up when several attributes are viewed together, which is easy to overlook by eye. Our project therefore adds a layer of behavioural analysis on top of the normal access-control process. Instead of asking only whether access is permitted, it looks at the circumstances around the request and uses machine learning to pick out combinations of attributes that may point to activity worth a closer look. The system takes in structured access-log data, validates and preprocesses it, derives behavioural features, and lets a Random Forest model label each event as normal or suspicious. Risk-based alerts are then assigned, and an audit history is kept so that everything can be browsed through a dashboard. Its role is to support administrators by pointing out events that deserve review. It does not decide on its own that a user is acting maliciously, and it does not take the place of login security, role-based permissions or access-control lists. What we offer in practical terms is a single prototype workflow that joins log processing, behavioural classification, risk indication, alert display and audit logging.

2. LITERATURE REVIEW

Access control is where most storage security starts. Role-Based Access Control (RBAC) hands out permissions through organisational roles, as described by Sandhu et al. in 1996 [1], while Access Control Lists (ACLs) attach a list of permitted users or groups to individual resources. Both help enforce least-privilege access and keep permissions

 This work is licensed under a Creative Commons Attribution 4.0 International License

manageable. Their job, though, is authorization. Neither tells us whether an authorised user's behaviour right now fits what is normal for that person.

Intrusion-detection systems take a different view. Scarfone and Mell explain in their NIST guide that such systems watch sources like network traffic, authentication records and system logs for signs of an attack [2]. They are good at spotting known patterns of misuse, but they are not built around the habits of an individual user.

Anomaly detection follows a related idea by flagging observations that depart from an established picture of normal behaviour, a field surveyed broadly by Chandola et al. in 2009 [3]. For file storage, the useful warning signs are fairly easy to list: access outside working hours, a source address never seen before, an unfamiliar location, repeated requests and contact with sensitive resources.

The same idea has been studied closely in network settings, where Ahmed et al. reviewed the main anomaly-detection techniques in 2016 [4]. Their survey is a reminder that detecting an unusual event and keeping false alarms low are separate problems, and the second matters just as much in a monitoring tool people have to use every day.

Insider threats are harder, because the activity often comes from an account that already holds legitimate permissions. Homoliak et al. reviewed the taxonomies and countermeasures in this area in 2019 [5], and a good deal of work has since looked at user behaviour, past activity and surrounding context.

One practical obstacle is the shortage of realistic data. Glasser and Lindauer addressed it by proposing a pragmatic way to generate insider-threat datasets [6], which makes it possible to test detection ideas without exposing real user records.

Tuor et al. later applied deep learning to structured security logs without relying on labels [7]. A point that runs through this literature is that an unusual event is a prompt to investigate and not proof of bad intent, since ordinary work sometimes involves odd hours, travel or a sudden jump in workload.

Machine-learning classifiers can weigh many structured attributes at once and pick up relationships that a person reviewing logs would struggle to see. Random Forest, introduced by Breiman in 2001, builds many decision trees and combines their outputs into one classification [8]. It suits the tabular access-log features used here and is readily available in standard libraries. The gap we try to fill is a practical one: putting log preparation, behavioural classification, risk-based alerts and audit review together in one compact prototype. We are not proposing a new classification algorithm. We want to show how these pieces can fit into a monitoring workflow for information storage.

3. METHODOLOGY

The process starts when file-access information is recorded or uploaded as a structured CSV file. A Log Collector receives the event details and hands them to the preprocessing module, and the cleaned features are then scored by the trained Random Forest classifier. Its result moves on to the risk and alert module, the event and its prediction are saved in the audit store, and finally the Streamlit dashboard shows recent activity along with the events that need an administrator's attention.

Depending on what the dataset provides, an input record may hold user and file identifiers, the type of operation, a timestamp, an IP address, a location, an access-frequency value and the sensitivity of the file. Validation confirms that the required columns exist and looks for values that are missing, inconsistent or invalid. Duplicate or unusable rows should be handled under a documented rule so the training data stays consistent. Feature engineering then turns the raw fields into numbers the model can use. A timestamp becomes the hour, the day of the week or a flag showing whether the access fell within working hours. Access frequency captures how many files a user requests in a given period, IP address and location add network and geographic context, and file sensitivity reflects how important or restricted the resource is. A user's earlier activity acts as a baseline against which the current event is compared. Categorical values have to be encoded the same way every time, and missing values dealt with before training or prediction, because the steps applied duringtraining must also be applied to every new record. Otherwise the features seen at prediction time would differ from those the model learned from.

Random Forest is the main prediction method. It grows a set of decision trees, each built from a different sample of the training data and a different selection of features, and for classification every tree casts a vote while the majority decides the class [8]. Here the two classes are normal and suspicious file-access events. The model learns from labelled access records during training, and when a new record arrives its processed features go to the trained classifier, which returns a class. We do not treat that output as a final security verdict. It is a pointer that helps the administrator decide which events need a closer look.

Predictions then pass to a risk-assessment stage. Normal events can simply be stored without raising a significant alert, whereas suspicious ones are highlighted for review, with risk levels such as Low, Medium and High assigned using criteria defined within the system. An alert can show the user or file reference, the access time, the predicted class and the risk level. Every processed event is also written to the audit store together with its prediction, risk level and alert status, so administrators can look back at earlier events, notice patterns that repeat and see why a given alert was raised. That trail is a useful source of records for later testing and for improving the model. The application logic is written in Python, with Pandas and NumPy handling loading, cleaning and numerical work and Scikit-learn supplying the Random Forest classifier [9]. The monitoring dashboard is built with Streamlit and is meant to present activity, predictions, risk levels, alerts and historical records in a form far easier to read than a raw CSV file.

4. RESULTS AND EVALUATION

4.1 Testing Approach

Testing has to cover both the data pipeline and the dashboard. Input validation can be tried with records that have missing columns, blank values, invalid timestamps or duplicate rows. Prediction tests should use representative normal and suspicious examples, and dashboard tests should confirm that recent records, prediction labels, risk levels and alerts appear correctly and that audit information can still be reached when reviewing history. A trustworthy evaluation of the model also requires the dataset to be split into training and testing portions [9]. The test records must stay apart from model fitting and from any preprocessing decisions, and the model should be judged on records it has never seen.
Useful measures include accuracy, precision, recall, false-positive rate and prediction time. Precision shows how many of the flagged events really are suspicious in the labelled test data, and recall shows how many of the suspicious examples were caught. The false-positive rate deserves particular attention, because too many alerts quickly make a monitoring system tiresome to use [4].

4.2 Current Results

The project report describes a working prototype workflow and a dashboard demonstration built on the demonstration data available. It does not include a complete, independent evaluation on a properly prepared test dataset. For that reason we give no numerical values for accuracy, precision,recall or false-positive rate, and we make no claim about how well the model detects suspicious activity. Functionally, the system links together access-record input, preprocessing, Random Forest prediction, risk-based alerting, audit retention and dashboard display. Our expectation is that several contextual signals taken together say more than any single field alone [6]. A request outside normal hours may be perfectly legitimate by itself, but the same request combined with an unfamiliar location, a high access frequency and a sensitive file may well deserve a closer review.

4.3 Limitations

How reliable the predictions are depends on the quality, size and representativeness of the labelled access records. If the data does not capture the normal variation among users, locations and working patterns, the classifier may give misleading results [3]. False positives are possible when genuine users work late or travel, and suspicious activity that looks like normal behaviour may slip through. The prototype is therefore best seen as a tool for supporting decisions, with administrators remaining responsible for investigation and response.

5. SYSTEM IMPLEMENTATION AND DEPLOYMENT

The system is divided into six linked components. The Log Collector takes in file-access records along with their contextual metadata. The Preprocessing Module validates the input, deals with missing or inconsistent values and prepares the features for the model. The Prediction Engine runs the trained Random Forest classifier from Scikit-learn [9]. The Alert Module flags suspicious predictions and attaches a risk level, the Audit Store keeps the event and prediction details, and the Dashboard shows the processed information to the administrator.

The workflow through these components has five steps. A file-access event is first captured with whatever metadata is available. The record is then checked and converted into the feature format the model expects, after which the classifier predicts whether the event looks normal or suspicious. Suspicious predictions move on to the risk and alert stage, and last, both normal and suspicious events are stored for auditing while the dashboard displays the relevant details. Keeping these stages separate means data preparation, prediction, alerting and review can each be changed or checked on their own.
The prototype can accept CSV records, report invalid input, preprocess the usable records, classify events, show risk information and keep a history of predictions. The scenarios we have in mind include an ordinary access event, one that happens outside normal working hours, access from an unfamiliar location, access to a sensitive file, and a case where several unusual signs appear together. For the unusual cases the intended response is extra scrutiny, not an

automatic assumption that each one is an attack.

For deployment, the prototype runs as a Streamlit application that an administrator can launch on an ordinary machine and point at a CSV export of access logs. Moving it into a live storage environment would call for a continuous log feed and the larger, independent evaluation described in the previous sections.

6. DISCUSSION

6.1 Behavioural Context alongside Access Control

The design shows how machine learning can bring behavioural context into conventional file-access security. Authentication confirms who someone is and authorization checks what they may do [1], while behavioural prediction asks whether the way they are accessing files looks unusual. These are different questions, and the layers work best when treated as complements to each other.

6.2 Traceability and Interpretation of Risk

The dashboard and audit history matter just as much. A prediction is of little use if an administrator cannot tell which event produced it or see the activity around it. By keeping each event together with its prediction, the system supports traceability and gives a base for later analysis. Risk labels should also be read as ways of prioritising attention and not as firm statements about what a user intended [5].

6.3 Evaluation Needs

A thorough assessment should look at how the model performs across different user groups and event types rather than relying on one overall score. Missed suspicious events and false alarms carry different operational costs, so they should be considered separately [4]. Before the system is considered for a real storage environment, it needs to be tested on a larger, independently prepared dataset, and the data source, labelling process and feature definitions need to be documented.

7. CONCLUSION

In this paper we presented an AI-based file-access prediction system for secure information storage. The workflow studies contextual features from access logs with a Random Forest classifier [8] and links each prediction to risk assessment, alert generation, audit logging and a Streamlit dashboard. It is meant to work alongside existing authentication and authorization mechanisms by drawing attention to access events that may need further investigation. We make no quantitative performance claim, since a complete independent evaluation has not yet been carried out. The next step is to prepare a representative labelled dataset, define a reproducible train-test procedure, and measure accuracy, precision, recall, false-positive rate and prediction time. Future work could include real-time log monitoring, cloud-storage integration, explainable predictions, larger-scale testing, alternative anomaly-detection methods and carefully controlled integration with security-monitoring platforms. Because user behaviour and storage environments keep changing, the system will also need continuous evaluation.

REFERENCES

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[2] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication 800-94, 2007.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, Art. 15, 2009.

[4] M. Ahmed, A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques,"*Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[5] I. Homoliak et al., "Insight into Insiders and IT: A Survey of Insider Threat Taxonomies, Analysis, Modeling, and Countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, Art. 30, 2019.

[6] J. Glasser and B. Lindauer, "Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data," in *Proc. IEEE Security and Privacy Workshops*, 2013, pp. 98–104.

[7] A. Tuor et al., "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams," arXiv:1710.00811, 2017.

[8] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

 This work is licensed under a Creative Commons Attribution 4.0 International License