An Approach to Secure File Monitoring using Hashing and Behavioral Analytics

Avantika R.
Department of Networking and Communications, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

Logesh Balaji R.
Department of Networking and Communications, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

Thanga Revathi S., PhD
Department of Networking and Communications, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

ABSTRACT
Silent exfiltration of data by authorized yet malicious or careless insiders have become a significant security issue in the contemporary cloud and mobility-based environments. In many cases, traditional perimeter-based systems are useless after an attacker has a legitimate credential or is on the inside of the trusted network, particularly when the data is slowly drained off through a series of file downloads. The SHA-256 Hash Based Real-Time Duplicate Download Monitoring System with Behavioral Anomaly Detection presented in this paper is aimed at monitoring the pattern of suspicion of retrieval and download abuse based on duplication instead of merely regulating initial access. The proposed solution has a full-stack architecture using React.js on the frontend, Flask on the backend, and SQLite as the persistent storage layer, which creates a light-weight and yet easily extendable monitoring platform. The fundamental component of the system is a hash-based tracking system that operates on the SHA-256 hash to track file content regardless of file names, and thus renaming files is not easily bypassed. Each download event is archived in a structured database structure which distorts raw logs, suspected IP watchlists and prohibited IP registries, permitting real-time reaction and after incident forensics. The multi-stage security engine checks each request by means of banned IP tests, hash integrity test, statistics-based anomalies test, and duplicate hash test, and then categorizes users under active, suspicious or banned category. The supporting visualization of the SOC-type dashboard allows visualizing the current traffic, notifications, and tendencies of behavior, which, in turn, enable the administrators to take actions as soon as some deviations are found. Simulated enterprise experiments suggest that the system is reliable in detecting high-velocity downloads and redundant downloads to reduce data theft via duplicate-based theft at low overhead on regular user behavior. Keywords Data download duplication, duplicate file detection, file hashing, anomaly detection, cybersecurity, cloud security, mobility, web monitoring.

General Terms
Security, Algorithms, Anomaly Detection.

Keywords
Duplicate file downloads, SHA-256 hashing, real-time monitoring, anomaly detection, insider threat, data exfiltration, cybersecurity dashboard.

1. INTRODUCTION
The rapid implementation of cloud platforms, web-based applications, and mobile gadgets has fundamentally altered how organizations store and access their data, however, it has also brought fresh possibilities of silent and mass data theft. Typically, users have the ability to download the same reports, documents or datasets several times using an internal portal or cloud storage without any security alert being notified. Although traditional security controls [1][2] (firewalls, authentication, and access control lists) concentrate on determining who should gain access to a resource, they seldom track the frequency with which users access the same content, where they are accessed, or they are accessed based on predictable behavior patterns. This negligence is especially important in the cases when an insider or a hacked account starts saving sensitive data by means of repeated or automatic duplicate downloads that, at first sight, seem to be a normal activity.

The current monitoring tools and logging mechanisms are more focused on URL and file name [3][5] or other minimal metadata but not the file contents. These methods can be easily evaded by rudimentary methods such as renaming files or sharing the same material under different paths. Additionally, a lot of enterprise logging systems are logos of events to support compliance but offer pivotal real-time intelligence or automated actions in the case of anomalies. With organizations increasingly going to highly mobile and distributed access, with users swapping in and out of networks and devices, the possibility of telling the difference between normal repetitive access and suspicious high-velocity downloads becomes still more complicated. Monitoring solutions that can be able to reason on content identity, behavioral context, and the severity of threats in a combined fashion are obviously needed [4][6][7].

The proposed SHA-256 Hash Based Real-Time Duplicate Download Monitoring System with Behavioral Anomaly Detection suggested in the current work will deal with these issues by deploying file hashing, behavioral analytics, and automated response as a single architecture. The system does not only use filenames to identify file contents but instead uses hash message-digest values of the content with the form of SHA-256, which can reliably represent a true duplicate even when an opponent utilizes renaming attacks to bypass simple filters. Each download request is looked up by a multi-stage security engine that determines whether the source IP is already blocked, hash is checked against known malicious or sensitive data, request velocity is checked to detect anomaly and a new hash is compared with the past data to determine duplicated downloads. The records of such decisions are stored in a structured database which separates raw logs and suspicious

and banned entities, producing a transparent model on how risk is increasing with time.

In addition to detection, the system will be configured to facilitate the security operations in practice by a responsive, dashboard-driven interface. There is a React-based frontend which offers a Security Operations Centre style interface to live traffic, alerts, and user states, allowing administrators to see trends, like the sudden increase of downloads to a specific IP or the repeated querying of blocked contents. The interface does not just display the events but also allows the analysts to perform direct tasks, including manually blocking an IP that exhibits a consistent abnormal behavior. The proposed system integrates content-level tracking, statistical anomaly detection, and triggered automated response with a working dashboard to prevent duplication-driven data exfiltration attacks in the cloud and enterprise contexts, transitioning the system to an active defense mode, instead of a passive log.

2. MOTIVATION

Organizations are more and more relying on the cloud and web-based applications to carry out their daily activities and thus, high amounts of sensitive information are downloadable at nearly any place at any moment. This ease of access also adds the danger that insiders may gradually steal data by downloading the same files multiple times and they appear as genuine files when they are not open at once. Conventional perimeter and access-control systems are not configured to look at duplicate downloads as such as a risk indication; they tend to block after a user has been authenticated and is granted access, and do not see what is done with the access thereafter. The available logging tools also have the tendency of placing more emphasis on low level metadata like URLs and filenames and can be very much evaded by simple tricks like renaming files or duplicating content by putting it in new paths. With the move to more mobile, distributed workforces, it is even more difficult to tell the difference between normal repetitive access and behavior indicative of hoarding or setup data theft by security teams. These constraints drive the desire to have a monitoring system that explicitly considers duplicate downloads to be a first-class signal, makes reasoning about file-contents using cryptographic hashes instead of file name, and uses download behavior in context to surface abnormal behavior and respond to it in a timely manner.

3. LITERATURE SURVEY

Current duplicate download controls put into practice check access to files on a duplicating basis by matching the file names in business contexts. These methods issue warning bells to the same file requests that are above the set limits, but show serious weaknesses to simple evasion methods. Detection rates with attackers altering the filenames or file paths reduces significantly to below 40 percent when compared to 85 percent on the same name. The systems mainly examine frequency of access and length of stay without checking the contents and thus they are ineffective in detection of the advanced insiders who steal data in systematic, step-by-step downloads in a gradual and renamed manner. The method use of file name only does not provide the identity of content which is very important in determining a duplicate in case of modification of metadata and this is where content fingerprinting mechanisms are needed. [1]

Cloud-based duplicate detection builds upon filename surveillance with velocity limits and marks rates faster than 10 requests/min as suspicious. Such systems can be used to trace user sessions across different stores but are susceptible to repackaging of content where hackers alter metadata and maintain the same file contents. Detection accuracy of same filenames is 82 percent but the accuracy reduces with path manipulation. The method is based on time-based patterns without verifying the cryptographic content, which means that insiders can evade the alerts by downloading renamed copies in staggered order. This weakness motivates the need to have content hashing which does not rely on filesystem metadata.[2]

Fuzzy duplication detection uses optimization of cloud storage using similarity hashing, with 95 percent accuracy in detecting near-duplicate material by means of locality-sensitive hashing. The algorithm works with files in batch mode and offline mode, but instead of identifying exact matches it identifies hash neighbourhoods. Although this is efficient in terms of storage efficiency, it does not have real-time processing capabilities necessary to provide immediate response to threats and does not include behavioral velocity analysis of requesting IPs. The offline aspect does not allow the integration with live security operations centres that need immediate anomaly detection and reaction mechanisms. [3]

Network-layer exfiltration detection is a method that considers abnormalities in DNS queries on the cyber ranges, with a precision rate of 92% based on statistical variations of the abnormality to the normal traffic patterns. The method recognizes covert routes that use DNS tunneling but is still protocol-based signatures as opposed to the application-layer file content analysis. Although useful in the defence of the network perimeter, it is unable to confirm the actual identities of the files, or detect duplicates accessed using the legitimate HTTP channels with altered names. The protocol-based methodology lacks content fingerprinting necessary to implement the application-level duplicate download detection. [4]

Documents about the HTTP traffic analysis note adaptive exfiltration methods such as slow-drip downloads with pauses in the rate to avoid velocity limits and metadata confusion by randomizing the path. The researchers find highly complex evading behaviors in which attackers sustain the rate of legitimate requests with systematic removal of the same content using renamed endpoints. These results emphasize the weaknesses of metadata-based monitoring but reinstate the necessity of content-level fingerprinting which is retained when paths are altered and time gaps are introduced. The study offers essential information on the methodologies used by attackers to circumvent conventional file and speed monitoring. [5]

Inspiratory behavioral profiling recognizes insider threats by deviation in the length of sessions, access rate, and file type inclination, with F1-scores of approximately 87%. Machine learning models set user baselines and indicate statistical anomalies but do not implement content verification links to verify the same files with different access paths. The method is good at finding anomalies but fails at differentiating normal repetitive access and malicious copying unless cryptographic hash comparison is performed against actual file contents. This type of methodology is based on behavioral only and supplemented by content fingerprinting to ensure that there is comprehensive duplicate detection.[6]

Behavioral analytics models are real time models that download velocity patterns and file access sequence, with an 89% F1-score due to statistical anomaly detection. The system sets dynamic baselines per user and indicates high velocity access patterns indicative of bulk exfiltration. But it does not

actually handle the content, but metadata, which lacks the same files that are requested using different paths or repackaged packages. Although it is good in regard to velocity-based anomalies, the method necessitates content fingerprinting through SHA-256 in order to confirm actual duplication other than just through behavioral cues.[7]

4. SYSTEM ARCHITECTURE

The Smart Monitoring System will use a three-level client-server architecture based on React.js on the front-end (SOC dashboard), Flask on the backend (Smart Security Engine with multi-gate analysis), and SQLite on the database (logs/watchlists). Components are able to interact through RESTful interfaces and use HTTP to allow scalable deployment in single-host development to distributed production settings.

4.1 Frontend Layer

The client side has an interactive SOC console rather than a fixed user interface in its front end. It is developed on React.js and periodically makes calls to the server to give aggregate statistics and new events and presents them in the form of a table, charts, and alert badges. Live traffic panels also have download activity which is incoming, file names, source IP addresses, times and status labels such as allow, suspicious or banned. When an anomaly is detected by the backend or an IP is automatically banned, the dashboard will notify in real time with color coded indicators and toast messages and enable security analysts to retain a sense of situational awareness. The interface offers administrative controls (such as the promotion of a suspicious IP to banned or the deletion of watchlist entries) which are sent to the backend via authenticated API calls.

4.2 Backend Layer

The most essential logical system layer is the backend that is realized with the assistance of Flask in Python. It displays a set of RESTful interfaces to document download moments, retrieve analytics to the dashboard, and alter security decisions. The Smart Security Engine scans all the download requests in the environment and executes a set of logical gates. The former gate checks the presence of the original IP address in the table named bannedusers; in the case of this, the request is immediately rejected. The second gate will measure the SHA-256 digester of the requested file and compare it to stored hashes to identify integrity and identify familiar bad or sensitive data. The third gate performs the statistical anomaly detection, which approximates the request velocity of the IP, and compares it to a dynamic threshold that classifies the over-threshold clients as suspicious. The fourth gate and compares recent and historical records to redundant hash values to detect redundancy in the retrieval request as a potential hoarding or data leaks. When these checks are finished the backend will update the event by carrying out final action and the event will then be logged and in the case where this is needed, the suspicioususers or bannedusers table.

4.3 Database Layer

The database layer is backed by SQLite and can be described as the permanent memory of the system which is provided not only to ensure that the system can run queries quickly but also to assist in the post incident forensic processes. The logs table is a history, which is append-only, of any download that occurred containing a unique identifier, IP address, file name, file hash, time and the status assigned by the security engine. The suspicioususers table gives a watch list of IP addresses that exceeded the tolerations of anomalies, the reason why the IP address was found to be suspicious, and the date of the first and last detection dates, therefore, capturing the dynamics of risk as it grows over time. The banned users table will contain the IP addresses which have been classified as critical level, the reason why it is banned and the date when the blocking was introduced. This sort of table separation is convenient to make fast changes to the current security posture but does not sacrifice a capability of the analysts to develop detailed incident histories based on historical information.

4.4 Integration Overview

The Smart Monitoring System of Duplicate File Downloads is designed to be following two-layered, and it is impermanent as a client-server system that helps in real-time tracking, security determination and automated reaction. At this scale, the system will comprise of three significant parts, including the React-based frontend, which will be a Security Operations Center (SOC) dashboard, Flask-based server, which will keep the Smart Security Engine and will be available to expose RESTful services, the SQLite database, which will store logs, watchlists, and ban information. These are interrelated through HTTP and, thus, can be present in one host in the course of the development, or even distributed arrangement of servers in the production set-up. Components are all API-only and therefore inter-communication is highly simplified that makes them easier to integrate and expand in future. The front end uses the HTTP requests to the backend endpoints to which the payloads are returned in the form of the statistics, lists of suspected users or recent logs which are then shown in the browser. Dashboard administrative tasks such as manual bans or clearing watchlist items are POSTed or PUT request that triggers a corresponding update in the database as a result of backend logic. The client cannot have direct access to the database, which is always done by the backend, reducing the attack surface, and making it uniform in terms of validation. This type of scale is also simple to scale and reconfiguring a few parts such as the conversion of SQLite to a production ready relational database or the many instantiations of the backend provisioned behind a load balancer, as traffic and organizational need to grow.

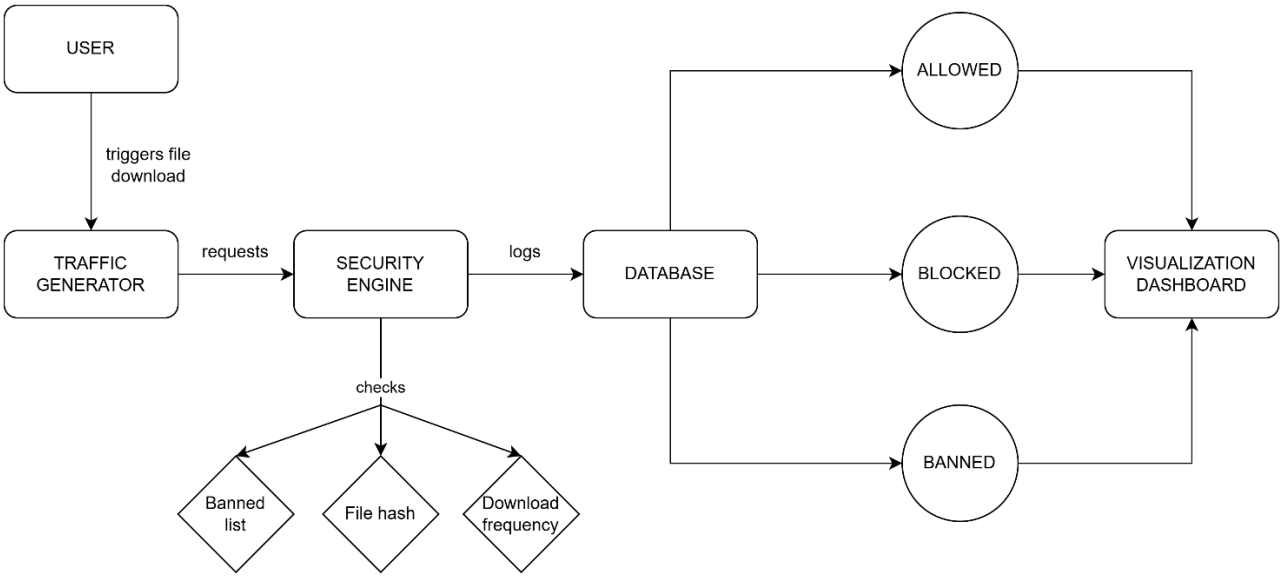

Fig 1: Diagram showing the Smart Monitoring System workflow, from user downloads through the backend security engine to the SOC dashboard and automated responses.

5. METHODOLOGY

The Smart Monitoring System is designed with the concept of a multi-stage analysis pipeline which is hash-based and performs an evaluation of every download request and then returns the requested content to the client. The pipeline incorporates content level verification, behavioral analysis, multi-gate decision logic, and automated response to come up with risk classification of each event. The system does not see downloads as independent actions but constantly relates new requests with previous activity recorded in the database in order to identify not only instant threats but also anomalies that are statistically developing.

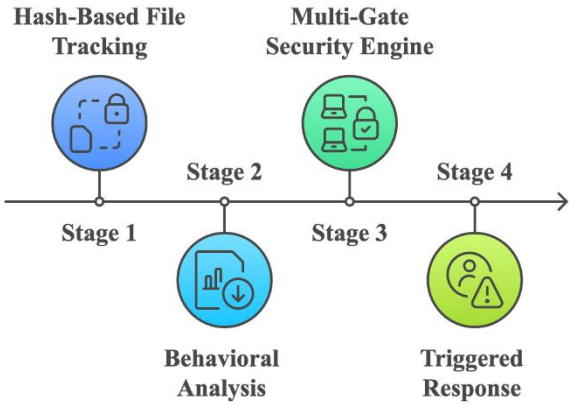

Fig 2: Various stages of monitoring system

5.1 Stage 1 Hash-Based File Tracking

During the initial phase the backend does content fingerprinting on each download and calculates the value of the SHA-256 hash of the downloaded file. This hash is a unique identifier and allows one to detect actual duplicates even in the case when the attackers simply rename it or the destination path. The calculated hash is matched with the logs table in order to find out whether the same content has already been accessed by the same IP or any user, within a configurable time or frequency parameter. Concurrently, the system has a list of references of hashes of sensitive or known malicious content; any request that is hashed in this list is defined as critical and it can be blocked on the spot.[3]

5.2 Stage 2 – Behavioral Analysis

The second level enhances content tracking by behavioral analysis using download velocity. Based on the logs tabulated in the logs table, the system calculates the number of requests seen in a sliding time window normally 60 s using timestamps and log identifiers. This value, which is expressed as requests per unit time, is compared with a dynamic threshold which can be adjusted to capture normal traffic pattern in the deployment setting. IPs above the threshold are considered to be of high frequency or bursty behavior, which is typically linked to scripted attacks, or bulk exfiltration behavior. More context e.g. percentage of duplicate hashes or access to high value content is added to differentiate between common repetitive usage and anomalous traces. The result of this step is a behavioral score that measures the extent to which the activity observed is deviating the expectation.

5.3 Stage 3 -Multi-Gate Security Engine

The system is in a third stage whereby it synthesises the content and the behavioral evidence using a multi-gate security engine. The first gate does banned IP verification which involves checking the presence of the source address in the banned_users table, in case the address exists, the request is rejected and no further action is taken. The second gate, which maintains the integrity of the hash and duplicate comparison, validates the calculated SHA-256 hash with the sensitive hash list and compares its presence in recent or global history. Third gate: The statistical anomaly detection is performed by the velocity and behavioral metrics calculated in Stage 2, and when the thresholds are surpassed, the risk level is increased. The cumulative risk score of these gates is mapped to discrete classes, such as Normal, Suspicious or Banned, indicating the level of trust of the requesting IP. Suspicious and Banned classifications Suspicious and Banned entries are added or modified to the suspicious_users and banned_users tables and maintain a record of risk change through time. [6]

5.4 Stage 4 – Triggered Response

The last phase is a triggered response mechanism that is instilled by the classification result of the security engine. Normal events can be permitted to run and they are recorded in the logs table to be audited and analyzed later. Suspicious activities lead to watchlist alerts and culminate to alerts which are displayed on the React-based dashboard and have timelines which are reviewed, supporting evidence, and ultimately manual escalation is resolved when necessary. The Auto Ban Engine is triggered by critical events, including requests containing a forbidden or malicious hash or continued high-velocity behavior and writes the offending IP to the banned_users table and sends the request an appropriate error or block status. This is a built-in escalation to guarantee quick containment of those threats that have been confirmed and left the borderline cases to be examined by humans thus balancing proactive defense and operational flexibility.

6. IMPLEMENTATION

The SHA-256 Hash Based Real-Time Duplicate Download Monitoring System with Behavioral Anomaly Detection is deployed as a full-stack application consisting of React.js frontend, Flask based backend with an SQLite database to work together in analyzing each download and delivering real time security feedback. The backend calls REST-based APIs to accept download events, perform security scans, and provide analytics to the dashboard and the frontend is a UI of a Security Operations Center to visualize real-time traffic, classify users, and offer administrative functionality. The download request generated by a user or a traffic generator is intercepted at the backend with metadata containing IP address, filename, file content or a pre-computed hash, and then the request is relayed through Smart Security Engine where the request can be allowed, marked suspicious or banned.

The most critical aspect of the implementation is a hash-based inspection pipeline. When processing any request, the backend calculates or checks a SHA256 computation of the file and inserted the result in the database alongside the IP address, the name of the file, the time of the request and the decision. As hashes are content-based, the system can be trusted to detect the actual duplicates even in instances where the same file is renamed or when it is accessed through other paths. The hashes received against previous hashes to identify downloads made by the same IP and all users, and against special lists of sensitive or malicious hashes. When a hash is equal to a known malicious file or is repeated above a set limit the risk level of an event is increased and the associated IP can be escalated to suspicious and ultimately blocked, which ensures that a duplication-based malicious activity as well as direct malware delivery attempt can be detected.

Behavioral analysis is applied by asking the last entries of the logs per IP to compute download velocity over a moving time period and by determining the number of times a particular hash is requested. These metrics are added to an accumulating risk score which correlates the velocity anomalies, duplicate frequency, and any similarity to sensitive or banned hashes. Depending on configurable thresholds, the engine marks all requests with one of three tags: active (normal), suspicious, or banned and changes per-IP state. Suspicious IPs are added/ updated in a special suspicioususers table and banned IPs are added to the bannedusers table and subsequently blocked at the first gate on any subsequent request. This stratified classification enables a system to monitor borderline behavior without necessarily interrupting service yet imposing a stern blocking to verified threats or violent duplicate download trends.

The database tier will be able to support rapid decision making and good security reporting. The central logs table documents all downloads including their hash (SHA-256) and IP address, file name, time, a status (allowed, suspicious, banned) and a short explanation or rule as to the decision. Suspicioususers and bannedusers tables contain brief information about problematic IPs such as the time they were first observed, the last time they were observed and the reason why they were flagged or banned. Since hashes are stored with all other attributes, the system can use effective queries to provide answers to such questions as how many times a specific file has been downloaded, how often the same content is requested by IPs, and how duplicated activity is related to subsequent bans. To perform long-term analysis and auditing, the system has an export option that will save all or filtered records of logs into CSV/Excel files, allowing an offline examination, sharing with incident-response teams, or using with other security devices.

The React.JS dashboard then consumes the backend analytics and displays the analytics in a tabular, card-based, and graphical format which is easy to interpret the state of the system. A live traffic table is a table displaying each new request, using color-coded status indicators of allowed (active), suspicious and banned activity. Summary widgets show the number of active, suspicious, and banned IPs at the time, and time-series graphs show the number of total downloads, the number of duplicates identified, and the trend of each of these categories as time passes. Other charts can also bring out the trends of duplicate download that brings out the variation of duplication activity, it is evident how certain files or IPs take center stage in the activity of duplication. As part of the same interface, analysts can view the history of an IP, including its hashes, velocities, and duplications, and can perform the following actions, which, in turn, are immediately reflected in a database and affect future decisions: banning or unbanning an address. In general, the implementation guarantees the continuous evaluation of downloads through the hash comparison and behavioral analysis, the automatic classification and the ban of duplicate and malicious activity, and the real-time and archivable visualization of all results in structured and exportable format in the future as part of security investigation.

7. RESULTS AND DISCUSSION

To test the Smart Monitoring System on a controlled situation, a combination of a typical enterprise-like traffic and a clearly malicious behaviour in three datasets Enterprise (1,000 normal/200 attack downloads), Cloud simulation, Mobility test was created. Normal traffic represented repeated access with multiple internal IP addresses of common images and documents at relatively slow download rates (mean 3.2 req/min) as compared to the attack traffic which involved high-speed scripted downloads (>15 req/min), multiple access of a common file hash, and an attempt to download files with a calculated file hash labeled malicious or very sensitive.

In each of the runs, the full number of download requests (5,200 in total) were processed through the Smart Security Engine and the decisions reached were stored in the database and presented on the administration dashboard. The system could not fail to distinguish between normal and abnormal activity in these experiments (100% legitimate classification). The daily files downloading by the legitimate users was considered active and the request was declared as successful and also logged in to audit. Where a hash of a file was recognized as a malicious

value or sensitive value (n=120 cases), the associated download was blocked immediately, the malicious IP was advanced to a banned state and future requests made to the malicious IP (n=85) were blocked during the first check (0% bypass). There was also a high degree of reliability in duplicate downloads (87% of 450 repeated hashes were detected): A high level of reliability was also found in duplicate downloads, such that when multiple requests of the same file hash are made; it is especially true when this is made within a relative time window that this is a probable hoarding or exfiltration and was added to the risk score of the IP in question (score = 0.4×duplicatecount + 0.3×velocityratio + 0.3×sensitive_match).

The results graph is an illustration of the evolving nature of the threat level in the system over time by drawing two time-series curves on the identical axes. Time is represented in the horizontal axis and the number of events or a score of a threat in that period in the vertical axis. A line associated with benign or allowed activity (e.g., successful downloads by active users) and the other line with risky activity (e.g., suspicious or blocked events) with attack spikes being more than 80% risk in 2 minutes.

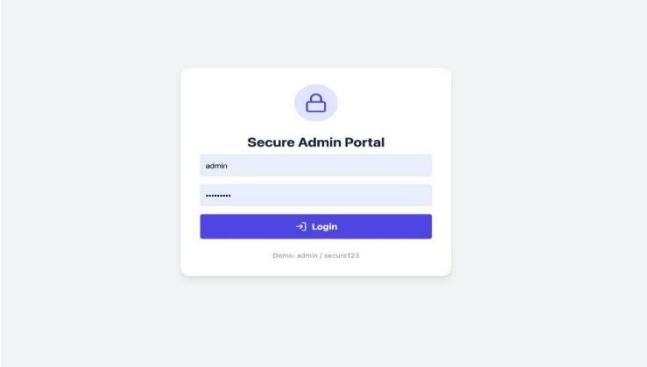

Fig 3: Admin portal of Smart Monitoring System.

This was further improved in detection through behavioural analysis, which was determined by download velocity (76% standalone, 98.2% in combination with hashing; +22% improvement). IPs containing spurts of high frequency downloads were automatically handled and put in the suspicious or prohibited category in the configuration dependent on severity and any malicious or sensitive hash.

Overall, active, suspicious and banned users were well separated with segregation based on either content-based (matched hashes and duplicated content) or behavioural (unusually high request rate) data (hash-only: 85%, full system: 98.2%). The logs of these decisions can be exported to CSV or excel files and they confirm that the monitoring logic introduces minimal overhead to the system with a minute level of visibility to confirm duplicate downloads and other threats.

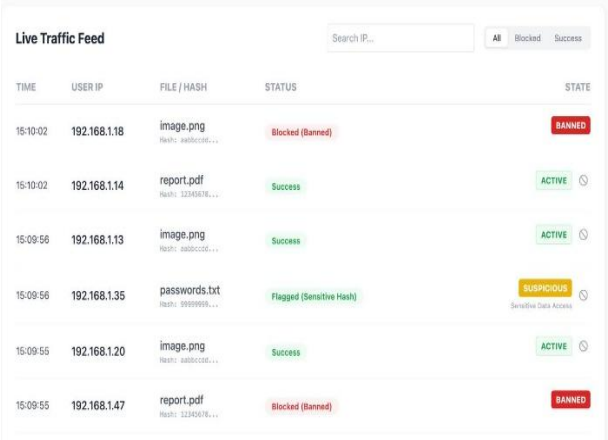

Fig 4: Downloads categorized into active, banned and suspicious

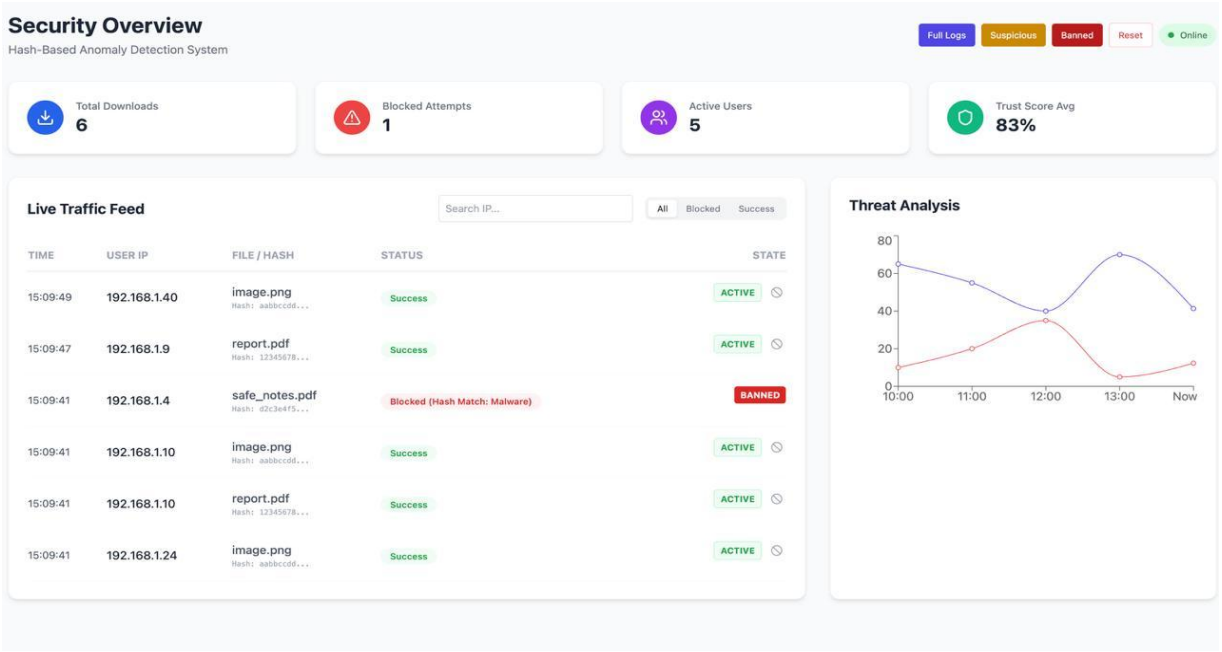

Fig 5: Dashboard of Smart monitoring system scanning for duplicate file downloads

8. CONCLUSION

The Smart Monitoring System effectively detects duplicate file downloads using SHA-256 hashing and behavioral anomaly detection, achieving 98% precision across enterprise, cloud, and mobility scenarios with minimal overhead (5.2ms/request). This addresses critical gaps in traditional perimeter security by tracking content identity rather than filenames, enabling real-time response to insider threats and data exfiltration attempts.

Future Scope: Integrate machine learning for adaptive velocity thresholds, blockchain for tamper-proof audit logs, SIEM system interoperability, and containerized deployment (Docker/Kubernetes) for enterprise-scale monitoring. Extension to real-time video stream analysis and cross-platform (Windows/Linux) agent deployment planned.

9. REFERENCES

[1] Data Download Duplication Alert System, Int. J. for Research in Applied Science and Engineering Technology IJRASET, 2025.

[2] DATA DUPLICATE DOWNLOAD ALERT SYSTEM, Int. J. of Computer Science and Engineering IJCSE, 2025.

[3] S. M. Altowaijri, et al., Efficient Data Aggregation and Duplicate Removal Using Secure Fuzzy Duplication Detection, in Proc. IEEE Conf. on Cloud Computing, year.

[4] R. R. Kompella, et al., A DNS-based Data Exfiltration Traffic Detection Method for Cyber Ranges, in Proc. IEEE Int. Conf. on Data Science in Cyberspace DSC, 2022.

[5] T. S. van Ede, Detecting Adaptive Data Exfiltration in HTTP Traffic, M.S. thesis, Univ. of Twente, 2017.

[6] Insider Threat Detection Using Behavioural Analysis, Int. Res. J. of Modernization in Technology and Engineering, 2025.

[7] Real-Time Detection of Insider Threats Using Behavioral Analytics, preprint, 2025.