SSHafe: A Real-Time SSH Brute Force Attack Detection and Novel Credential Rotation Standard

Aditya Mitra[1,3*], Amar Kumar Mandal[1,3], Amaan Rais Shah[1,3], Iqra Naz[1,3], E. Fatih Yetkin[2,3], Tuğçe Ballı[2,3]

[1*]CyberMACS.
[2]Department of Management Information Systems.
[3]Kadir Has University, Cibali, Istanbul, 34083, Turkey.

*Corresponding author(s). E-mail(s): aditya.mitra@stu.khas.edu.tr;
Contributing authors: amarkumar.mandal@stu.khas.edu.tr;
amaanshah@stu.khas.edu.tr; iqra.naz@stu.khas.edu.tr;
fatih.yetkin@stu.khas.edu.tr; tugce.balli@stu.khas.edu.tr;

Abstract

SSH remains a critical yet heavily targeted protocol for remote system administration, with password-based authentication exposing servers to large-scale brute-force, dictionary, and credential-spray attacks. Existing rule-based defences such as Fail2Ban fail to detect slow, distributed, or threshold-aware adversaries, while conventional account-recovery mechanisms: email links, OTPs, and out-of-band verification introduce additional vulnerabilities including phishing, session hijacking, and weak authentication binding. This work presents SSHafe, a real-time SSH brute-force detection and mitigation system that combines time-series feature engineering with a lightweight LightGBM classifier to identify attack patterns directly from system authentication logs. A multi-scale sliding-window approach extracts behavioural features such as attempt rates, inter-arrival times, failure ratios, and username diversity, enabling the model to achieve a detection accuracy of 99.96% on benchmark data and strong performance on unlabeled real-world traffic. Upon detecting an attack, SSHafe automatically blocks the targeted user account and delivers an SSH banner guiding legitimate users to a novel passkey-based password-rotation workflow. The proposed novel password reset standard performs authentication and password update in a single cryptographically bound flow, eliminating the need for sessions, cookies, OTPs, or email-based verification, and mitigating phishing, session hijacking, CSRF, and replay attacks. Experiments on an Azure VM and live adversarial traffic demonstrate that SSHafe can identify and suppress

brute-force activity within ten seconds, preventing account compromise even with weak credentials. By unifying real-time detection with a secure, sessionless credential-rotation mechanism, SSHafe provides a robust, end-to-end framework for defending SSH services against modern automated attack vectors.

Keywords: Authentication, Brute-force attacks, Credential rotation, FIDO2, Intrusion detection, LightGBM, Password security, SSH, Time-series analysis, WebAuthn

1 Introduction

The Secure Shell (SSH) protocol is the backbone of remote server administration, used daily across cloud platforms, enterprise infrastructure, and embedded systems worldwide. SSH offers two authentication methods: public key and password. Public keys are cryptographically stronger, but they come with a practical catch: one can only log in from a machine that already holds one's private key. Passwords, on the other hand, let one connect from anywhere, which is why they remain the more common choice despite the security trade-off. That trade-off is significant: passwords are a natural target for brute-force and dictionary attacks, in which automated tools cycle through thousands of credentials per second until one works, granting the attacker immediate access to the system. The standard defence, often implemented by tools like Fail2Ban, is to block an IP after too many failed login attempts, which sounds reasonable. However, it is easily defeated by attackers who simply slow down or spread their attempts across multiple addresses, staying just below the detection threshold [1]. This gap between real-world attack behavior and what rule-based systems can detect motivates the present work.

Password brute-force attacks are well-characterised zero-knowledge threats [2], yet existing account recovery mechanisms: email magic links, one-time passwords, and out-of-band SMS codes introduce secondary vulnerabilities including session hijacking and phishing [3]. A fully effective response, therefore, requires not only reliable detection but also a cryptographically robust, end-to-end credential rotation mechanism.

This study presents SSHafe, a real-time SSH brute-force detection and mitigation system. A machine learning model monitors authentication logs and, upon detecting an attack, immediately blocks the targeted account and alerts the user via an SSH banner with a link to a password reset portal. The user can then securely rotate their credentials through a novel passkey-based reset workflow, which binds authentication and password change into a single phishing-resistant flow, eliminating the vulnerabilities inherent in conventional recovery mechanisms such as OTPs and email magic links.

The main contributions of this study are:

- A machine Learning time series model to detect SSH brute force and password spray attacks.
- Dynamic configuration of SSH to block accounts facing the brute force attacks.
- A novel cryptographic approach for password rotation for the blocked accounts using FIDO2 Passkey standard.

2 Literature Review

The intrusion detection problem has remained one of the prominent issues in network security, and the detection method has evolved from simple rule-based to machine learning based methods. This also applies to the Secure Shell (SSH) anomaly detection problem, as in this study. A study [1] provides evidence that Adaptive Boosting (ADA) Boosted random forest detected 40% more SSH brute force attacks than traditional rule-based methods like Network Measurement Analysis (NEMEA) detection. The false positive rate decreased from 0.94% to 0.45% while maintaining the accuracy of 0.94% & Area Under Curve (AUC) of 0.998 because Machine Learning (ML) based solutions were able to incorporate protocol-based features like packet length and inter-packet time, which were often overlooked by simple rule-based methods. Another line of study [4] evaluated ensemble-based classifiers such as Bagging, boosting, and Rotation Forest on 34 months of honeynet data comprising of 2.6 million packets, which resulted in a reduction of Type-I error (False positive rate) as low as 0.028% on packet-based analysis and Type-II error (False negative rate) as low as 0.012% for session-based analysis. Similarly, a study [5] used Convolutional Neural Network (CNN) based approach on the CIC-IDS 2018 dataset [6] for brute force attack detection, achieving an accuracy of 94.3%, a precision of 92.5%, and a recall of 97.8%, clearly outperforming the classical ML classifiers like Naive Bayes, Logistic Regression, and Support Vector Machine (SVM).

Moving on to the time-based machine learning model. A study [7] demonstrated that Long Short Term Memory (LSTM) model achieved an accuracy of 99.88% on the CICIDS2017 dataset [8] for the SSH Brute Force detection task, outperforming a non-temporal-based machine learning model. [9] proposed an LSTM based system called Brute Force Secure Shell (BruSSH) for early detection of distributed brute force using sequential login failure counts, on two different datasets, i.e., SSH Traffic Dataset [1] and PANDAcap [10]. Their results outperform those of[1] in key performance metrics, achieving a recall of 99.7, a precision of 99.98, and an F1-score of 99.89. Another study [11] compared three different temporal-based ML models, LSTM, Gated Recurrent Unit (GRU), and 1D CNN, on a synthetic dataset, where it was found that GRU achieved the highest F1 score and accuracy amongst all. Similarly, Lee and Lee [12] combined feature engineering (inter-packet arrival time) with data augmentation (WGAN-GP) to improve the F1-score of SSH detection, achieving an impressive score of 0.999. All these SSH-specific collective studies demonstrate that the ML-based approach is substantially superior to classical rule-based approaches, although direct comparison between time-series specific architecture and well-engineered ensemble methods on identical SSH datasets remains unexplored.

The application of time series analysis is not just limited to SSH but also widely used in intrusion detection. The classical AutoRegressive Integrated Moving Average (ARIMA) model is used in volume anomalies detection [13], whereas the autoregressive part is with sequential hypothesis testing to identify abrupt changes [14] in network behaviour. A study [15] proposed a hybrid framework that combines the ARIMA model with a decision tree classifier. They used an ARIMA pipeline to filter out noisy traffic and reduced the packet rows from 30,000 to just 400, making the job of downstream classifier easy to identify attacks, improving the F1-score to 99.71% while retaining

95.8% attacks in the filtered rows. In contrast, a matrix profile algorithm (unsupervised learning) achieved an accuracy of 99.6% & F1-score of 99.8% on UNSW-NB15 [16] and CICIOT2023 dataset [17].

On the other hand, deep temporal models, particularly the LSTM variant, have shown strong performance on the canonical intrusion benchmark, often achieving 99% accuracy on the NSL-KDD [18] and CICIDS2017 datasets [8] , [19]. A hybrid approach, such as CNN-LSTM, which combines local pattern extraction with temporal features, has achieved 99.78% accuracy on the NSL-KDD [20] dataset. Similarly, studies by Q. Zhu [21] use a graph-based neural network approach, which models relational interaction between network entities and is reported to have achieved 99.99% accuracy on the benchmark dataset.

One of the most common feature engineering techniques for network data containing logs is the sliding window technique, which extracts important feature vectors necessary for a supervised machine learning model by sliding over the datasets [13], [14], [15]. Features such as attempt rates, failure rates, unique user count, and Inter Arrival Time (IAT) means are computed over multi-scale windows, which help capture different attack patterns from intense to prolonged anomalous behavior [1], [14]. Several studies support the claim that aggregation-based feature engineering techniques improve classifiers' sensitivity to detect an attack behavior [22], [23].

Tree-based ensemble methods, such as random forests, outperform other models and architectures when working with a structured intrusion detection task by achieving a competitive accuracy of 99.88% and an F1-score of 97.6% while also offering high interpretability and low computational cost during training [23]. In contrast, gradient boosting frameworks such as Light Gradient Boosting Machine (LightGBM) are more suitable for heterogeneous tabular features due to their ability to work with mixed feature vectors and learn non-linear decision boundaries [15] [23].

The existing literature reveals that most work focuses either on deep learning based architecture, such as LSTM [7], [9], [11], and CNN [5], or classical ML classifiers trained on non-temporal per connection-based features [1]. In addition to it, the majority of prior work did analysis using network-level packet data, such as flow records [1], [4], [5], [12], rather than application logs from the system, which provide direct access to ground truth like login outcomes, usernames, and authentication methods. Moreover, the existing literature predominantly focuses on detection in isolation, with little attention given to real-time mitigation strategies.

The proposed work addressed this gap by combining time-series analysis of the authentication log (/var/log/auth.log) with a lightweight machine learning model as a downstream classifier to detect attacks before they escalate. In addition, our proactive approach focuses on a real-time mitigation strategy that detects attacks and sends an alert to the user to take action and secure their account before the attacker can exploit weak authentication credentials. Table 1 compares the performance of the models.

One of the most important aspects of a successful attack detection pipeline is the ability to gain insights and take real-time decisions to secure systems, often critical, in the face of attacks from various types of threat actors. For systems involving passwords, the major threats involving zero-knowledge attacks are brute force, dictionary attacks,

Table 1 Performance Metric Comparison

Metric	SSHafe	BruSSH[9]	[1]	CNN [5]
Recall	**99.96%**	99.70%	99.49%	97.80%
Precision	**100.00%**	99.98%	99.38%	92.50%
F1-Score	**99.98%**	99.89%	99.43%	91.80%

Note: Bold values indicate the best performance.

etc. [24]. And one of the most proactive ways to mitigate such risks is password expiration and rotation. One of the most widely used methods for account recovery after an expired password is using recovery codes or links over emails, SMS-es, other out-of-band factors, etc. A study [25] discussed the vulnerabilities of such account recovery flows and offers a multi-factor Secure Email Account Recovery (SEAR) protocol involving SMS and e-mails. Another study [26] proposes one-time rotating passwords that are generated on a smartphone app. A study comparing passwordless authentication methods with FIDO2 against other authentication methods [27] highlighted the usability of FIDO2 against other account recovery channels.

However, all these account recovery methods often authenticate the user first over methods like email magic links, passwordless FIDO2, or other factors, and then initiate the password reset flow. This creates a binding issue between the one who authenticated and the one who is entering the new password. This is often dealt with using approaches like session cookies, JWTs, and more. However, a study [28] shows these cookies and tokens are often not expired or scoped well, which allows further account takeover after recovery. Another study [29] shows Cross-Site Request Forgery (CSRF) and other forms of attacks that may allow attackers to take over the authenticated session during account recovery. To mitigate these issues, this study proposes a novel FIDO2-based password reset flow that performs the authentication and password recovery in a single flow, without involving session cookies, tokens, or other approaches requiring a long-lived authenticated session. Table 2 compares SSHafe Password Rotation flow with other commonly used methods.

3 Methodology

The proposed study uses a machine learning model trained separately and then deployed on a cloud virtual machine (VM) for mitigating attacks.

3.1 Training workflow

A labelled dataset [30] on SSH information was used to train a Light Gradient Boosting Machine (LightGBM) model. The dataset had columns timestamp, source IP address, Username, event type, status, and label. The dataset had 90% Brute force records and 9% normal or non-anomalous records. While this looks like an unbalanced dataset at first glance, it resembles real-life data. The high count of brute force data can be attributed to the nature of brute force attacks, which, by definition, implies a malicious

Table 2 Security Comparison of Authentication Approaches

Approach	**Vulnerable to Email theft**	**Vulnerable to Brute force or dictionary attacks**	**Vulnerable to session hijack**	**Vulnerable to Social engineering**
Email magic link	Yes	No	Yes	Yes (Like CSRF after session is authenticated)
Old password	No	Yes	Yes	Yes (Like Phishing)
OTPs	No	Yes	Yes	Yes (Like Phishing)
SSHafe Password Rotation	**No**	**No**	**No**	**No**

actor trying random passwords very fast. Hence, the number of failures and brute force labelled data is naturally very high.

However, transforming this data to time domain data per user with a sliding window of 60 seconds effectively normalizes the dataset. This is because a high number of failed attempts to login to a particular user account in a short window of around 60 seconds is represented as a single attempt to brute force.

An exploratory analysis of the dataset was performed to determine whether it was fit for the purpose of this study. Raw count of both brute force records and normal access records was plotted along with their 60s rolling mean. Brute force attacks showed a high spike in very low time span to around 3 to 5 hits per second. This aligns with the definition of brute force and dictionary attacks where the attacker attempts to spray a high number of random passwords or from a wordlist in a very short amount of time. On the other hand, normal connections usually had only one successful connection or failure per second, with up to a maximum of two per second. This aligns with the general connection pattern where the user takes time to enter the password or does not attempt multiple connections very fast. This creates a very visible baseline for further analysis of the dataset. Figure 1 shows the temporal features of the dataset and the baseline difference between the count of attempts of a particular type (benign and attack) per second. It also shows a 60-second rolling mean for comparison.

A count distribution plot made it clearer. This shows brute force attacks have a wider range of intensity where the brute force attack often reaches 3 to 5 attempts per second, as compared to normal connectivity that is usually only one event a second. Brute force attacks have a high frequency compared to normal connections and are a signature of automated tools or scripts trying to authenticate quickly. Figure 2 shows the distributional patterns of the SSH attempts in the dataset.

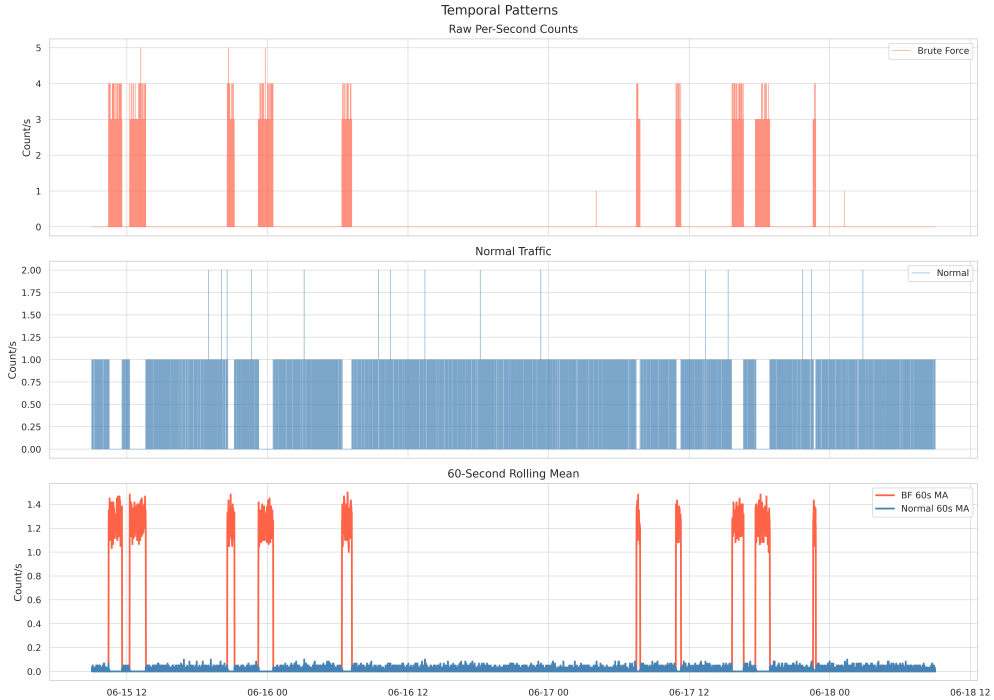


Fig. 1 Temporal patterns of SSH attempts

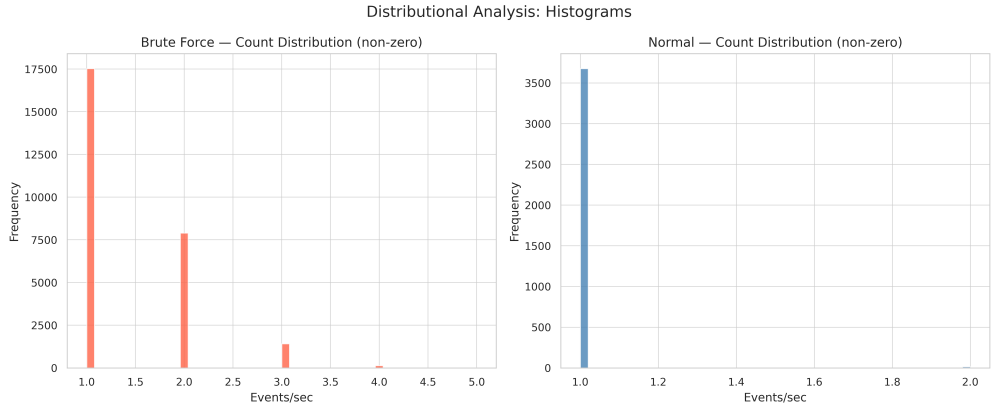


Fig. 2 Distribution patterns of SSH attempts

Stationarity tests were performed on the dataset. Two tests, Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests were performed. The results were contradictory, ADF test showed the dataset was probably stationary, while KPSS showed the dataset was probably non-stationary. This is interpreted as a valid result since the data is stationary. However, the given dataset showed attackers initiating the brute force attack periodically which could have influenced the KPSS

model to find out a non-stationary relation. Frequency-domain analysis on the dataset gave no actionable insight due to the absence of any harmonics.

3.2 Feature Engineering

Sliding windows of 30, 60 and 300 seconds were used to extract features like attempt rate, failure ratio, unique users, inter arrival time (IAT) mean and covariance, and number of attempts. The 30 second window was plotted for analysis, with the other windows showing similar results.

The attempt rate was considerably higher for brute force attempts. The failure ratio density for brute force attacks was around 50 while for normal attempts it was lower. Brute force attempts used higher number of usernames for username spray type attacks. The mean IAT for brute force attacks was considerably lower, and the covariance was higher. The number of brute force attempts in 30 seconds sliding window was around 38, while for normal connections, it was very low, around 1 or 2. Figure 3 represents the extracted features over a thirty second sliding window.

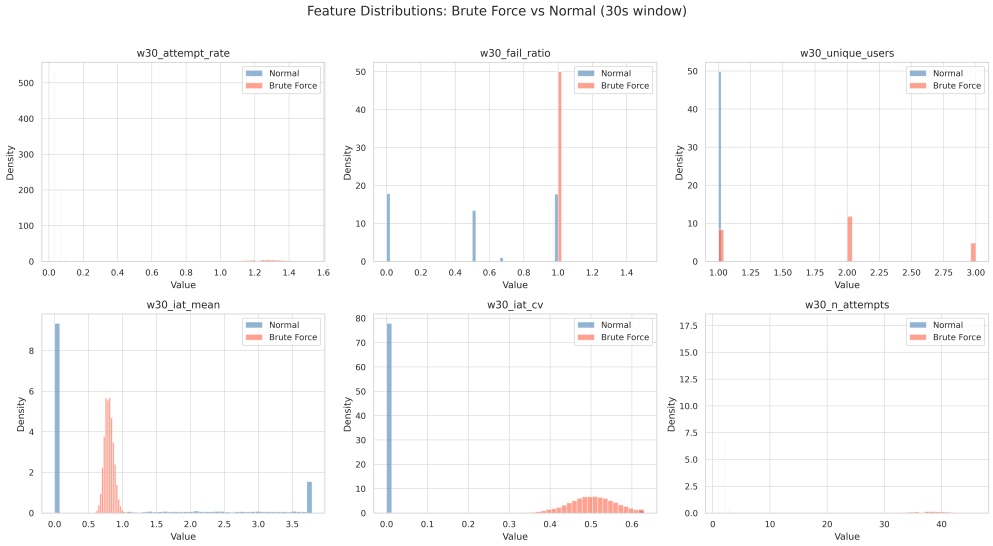


Fig. 3 Features extracted from sliding window

The features were considered, and a LightGBM model was trained. The accuracy was 99.96%, and the model was empirically validated against unlabeled real-world data with domain expert validation, reaching satisfactory performance. The system was found to focus on differentiating burst patterns of failed attempts to identify brute-force attempts.

The LightGBM feature importance list showed important features that matched empirical validation. The most important features considered by LightGBM were the number of attempts in 30 seconds sliding window, the number of failures in 30 seconds sliding window, the failure ratio in 300 seconds sliding window, and the mean IAT in

30 seconds sliding window. Figure 4 represents the importance of the features as used by the LightGBM model.

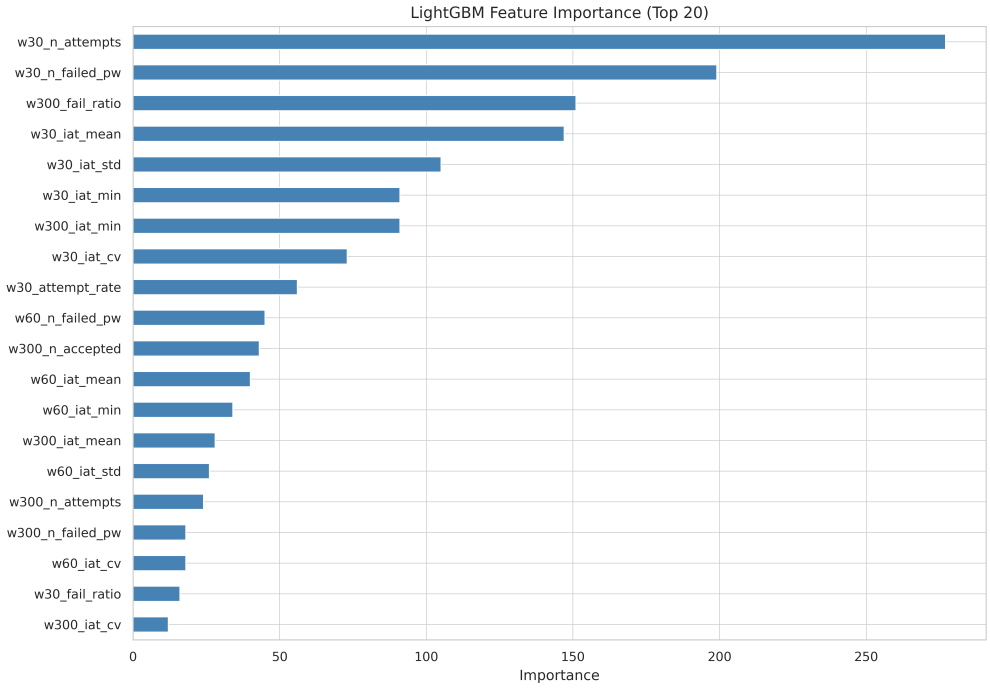


Fig. 4 Importance of features as used by LightGBM

The model reached high accuracy on the test dataset giving minimal false negatives and no false positives. Here false negatives imply the model predicted a request to be benign where it was a brute force request. This is not a problem, since brute force attacks depend on a large number of attempts in a short span of time, even if one is predicted as benign, the next one could be marked as an attack, thus taking the same actions. Figure 5 represents the performance of the model and shows the confusion matrix.

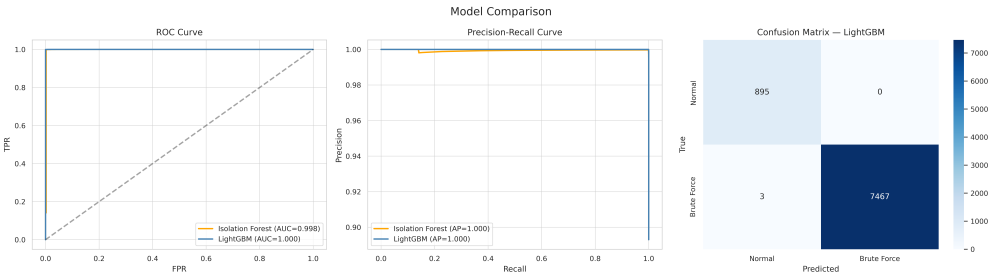


Fig. 5 Performance and Confusion matrix

Table 3 Per-Fold Cross-Validation Results

Fold	LGB AUC	LGB F1	ISO AUC	ISO F1
1	NaN	1.0000	NaN	1.0000
2	1.0000	0.9998	0.9915	0.9991
3	1.0000	0.9999	0.9996	0.9968
4	1.0000	0.9996	0.9993	0.9882
5	1.0000	0.9999	0.9950	0.9978

*AUC NaN: held-out partition contained a single class.

The threshold of 0.7 gave the maximum F1 score of 0.9998 and hence was chosen as the best threshold for using the model. The model was fine-tuned to use the best threshold for future prediction. Figure 6 shows the threshold of 0.7 giving the highest F1 score.

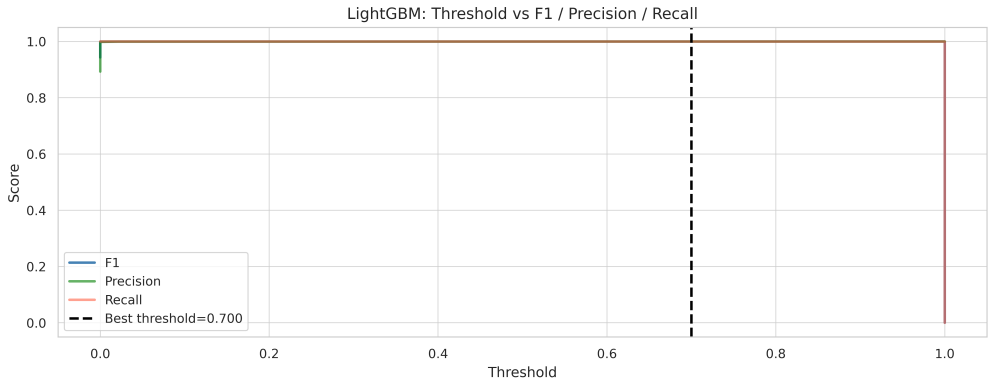


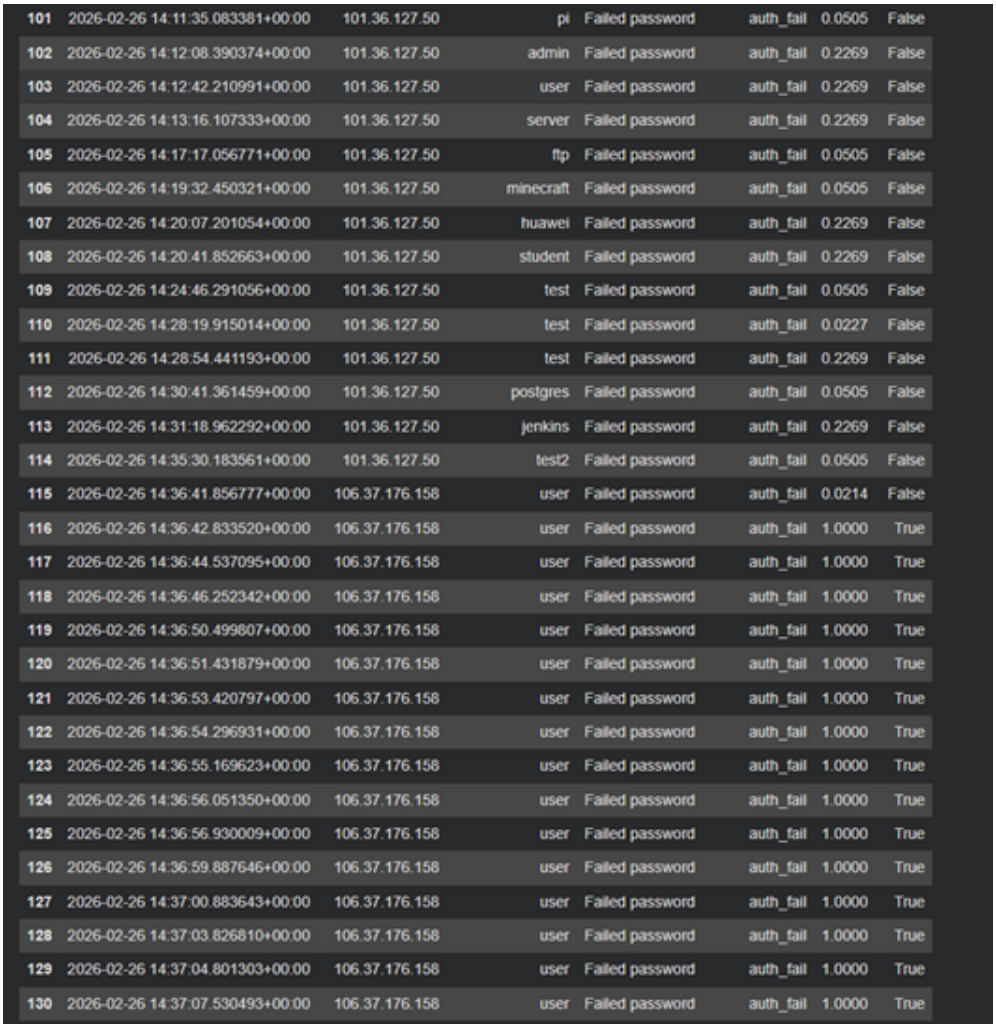
Fig. 6 Fine-tuning for best threshold

A conversion pipeline was developed to read the authentication logs (/var/log/auth.log) from a Linux machine and process it into an unlabeled dataset. Such data from 4 different cloud VMs were used to empirically validate model performance with domain expert knowledge. Figure 7 shows a section of the validation results of the model with logs from cloud VMs.

The model performance was satisfactory and deemed ready to be deployed for real-world usage.

3.3 Cross-Validation and Stability Analysis

To find out the generalization ability and the variance of the proposed detector, we have performed stratified 5-fold cross-validation on the labeled dataset. In each fold, LightGBM was retrained from scratch, and its performance was evaluated on the held-out partition. An unsupervised Isolation Forest trained on the same partition was used as a baseline model against which the performance of LightGBM was measured. Per-fold results are summarized in Table 3.



101	2026-02-26 14:11:35.083381+00:00	101.36.127.50	pi	Failed password	auth_fail	0.0505	False
102	2026-02-26 14:12:08.390374+00:00	101.36.127.50	admin	Failed password	auth_fail	0.2269	False
103	2026-02-26 14:12:42.210991+00:00	101.36.127.50	user	Failed password	auth_fail	0.2269	False
104	2026-02-26 14:13:16.107333+00:00	101.36.127.50	server	Failed password	auth_fail	0.2269	False
105	2026-02-26 14:17:17.056771+00:00	101.36.127.50	ftp	Failed password	auth_fail	0.0505	False
106	2026-02-26 14:19:32.450321+00:00	101.36.127.50	minecraft	Failed password	auth_fail	0.0505	False
107	2026-02-26 14:20:07.201054+00:00	101.36.127.50	huawei	Failed password	auth_fail	0.2269	False
108	2026-02-26 14:20:41.852663+00:00	101.36.127.50	student	Failed password	auth_fail	0.2269	False
109	2026-02-26 14:24:46.291056+00:00	101.36.127.50	test	Failed password	auth_fail	0.0505	False
110	2026-02-26 14:28:19.915014+00:00	101.36.127.50	test	Failed password	auth_fail	0.0227	False
111	2026-02-26 14:28:54.441193+00:00	101.36.127.50	test	Failed password	auth_fail	0.2269	False
112	2026-02-26 14:30:41.361459+00:00	101.36.127.50	postgres	Failed password	auth_fail	0.0505	False
113	2026-02-26 14:31:18.962292+00:00	101.36.127.50	jenkins	Failed password	auth_fail	0.2269	False
114	2026-02-26 14:35:30.183561+00:00	101.36.127.50	test2	Failed password	auth_fail	0.0505	False
115	2026-02-26 14:36:41.856777+00:00	106.37.176.158	user	Failed password	auth_fail	0.0214	False
116	2026-02-26 14:36:42.833520+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
117	2026-02-26 14:36:44.537095+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
118	2026-02-26 14:36:46.252342+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
119	2026-02-26 14:36:50.499807+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
120	2026-02-26 14:36:51.431879+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
121	2026-02-26 14:36:53.420797+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
122	2026-02-26 14:36:54.296931+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
123	2026-02-26 14:36:55.169623+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
124	2026-02-26 14:36:56.051350+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
125	2026-02-26 14:36:56.930009+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
126	2026-02-26 14:36:59.887646+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
127	2026-02-26 14:37:00.883643+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
128	2026-02-26 14:37:03.826810+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
129	2026-02-26 14:37:04.801303+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True
130	2026-02-26 14:37:07.530493+00:00	106.37.176.158	user	Failed password	auth_fail	1.0000	True

Fig. 7 Validation with unlabelled data

The aggregated statistics in the five folds, including the mean, standard deviation, and the 95% confidence interval (CI) for each metric, are presented in Table 4. It shows that LightGBM achieved highly stable performance with an F1-score of 0.9998 ± 0.0001 and accuracy of 0.9997 ± 0.0002. The isolation forest also performed strongly but showed a higher variance ($F1 = 0.9964 \pm 0.0047$). Since precision across all folds remained consistent (mean = 1.0000, std = 0.0000) for LightGBM, the confidence interval is undefined due to zero variance, indicating that no benign samples were misclassified. The mean recall of 0.9997 ± 0.0003 further confirms that very few brute-force events were missed. In contrast, the Isolation Forest baseline produced a wider spread, with the F1-score varying by almost half a percentage point between folds (std

= 0.0047), reflecting the greater sensitivity of unsupervised anomaly detectors to the composition of each fold.

Table 4 Cross-Validation Stability Report (Mean ± Std, 95% CI)

Metric	**Model**	**Mean ± Std**	**95% CI**
F1	LightGBM	0.9998 ± 0.0001	[0.9997, 1.0000]
Precision	LightGBM	1.0000 ± 0.0000	[1.0000, 1.0000]
Recall	LightGBM	0.9997 ± 0.0003	[0.9993, 1.0000]
Accuracy	LightGBM	0.9997 ± 0.0002	[0.9995, 1.0000]
F1	IsoForest	0.9964 ± 0.0047	[0.9905, 1.0000]
Precision	IsoForest	0.9929 ± 0.0093	[0.9814, 1.0000]
Recall	IsoForest	1.0000 ± 0.0001	[0.9998, 1.0000]
Accuracy	IsoForest	0.9942 ± 0.0068	[0.9858, 1.0000]

To further quantify the stability of our detector model (i.e., LightGBM), the model was additionally trained on 20 different random seeds $(7, 17, 27, \ldots, 197)$ while keeping the feature pipeline fixed. Across all the twenty runs, the model converged to identical metrics of AUC = 1.0000, F1 = 0.9998, Precision = 1.0000, and Recall = 0.9996, with no variance between the seeds. This conclude that the decision boundary learned by our LightGBM model is solely dependent on the discriminative temporal features extracted by sliding window rather than stochastic choices made during training which is an important characteristic for the application deployed in security critical infrastructure. Table 5 summarises the seed-based stability evaluation.

Table 5 Seed-Based Stability (20 Seeds)

Metric	**Mean**	**Std. Deviation**
AUC	1.0000	0.0000
F1-Score	0.9998	0.0000
Precision	1.0000	0.0000
Recall	0.9996	0.0000

The combined results from cross-fold validation and seed stability experiment provide evidence that our model has achieved high accuracy with minimal variance across all different data split and training run. This consistency makes it well-suited for real-time use on production SSH servers.

3.4 Attack mitigation

The study proposes a Daemon that would monitor the authentication logs and use the machine learning model to detect SSH brute force attacks. The system maintains a list of blocked accounts. Once a brute force attack is detected on a valid user account, the user account is appended to the list. This list uses a 'Match User' directive to find the blocked user and disable Password Authentication in SSH configurations. A SSH banner is added for these users showing password authentication was disabled and providing the web link to the password reset portal. For unblocking a user, the username is removed from the list, and the updated configurations are applied to SSH. The user account is automatically unlocked after a successful password reset. Figure 8 highlights the attack mitigation flow.

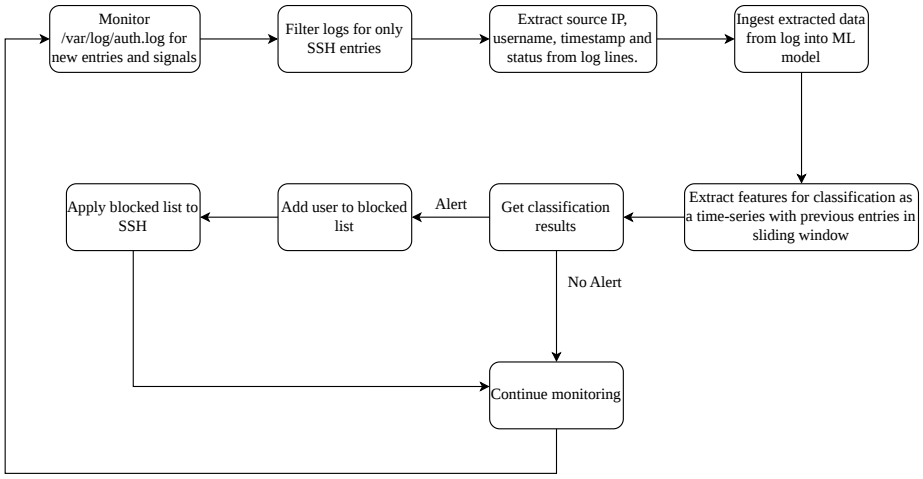


Fig. 8 Attack mitigation flow

3.5 SSHafe Password Rotation

This study proposes a novel password reset workflow that encapsulates passwords in passkeys to provide non-repudiation and integrity guarantees, while at the same time mitigating vulnerabilities relating to other forms of authentication before password reset, including OTPs, Tokens sent on emails, authentication with out-of-band credentials, and so on. Most password reset flows make authentication and entering password a two-step process, while involving other methods like cookies, JWTs, session storage, etc., to maintain the session. This introduces additional risks like cookie theft, session hijack, and so on. The proposed password reset workflow relies on phishing-resistant, passwordless authentication factors to carry the password and uses Web Authentication as an authenticated, encrypted channel. When a user account is created or modified, the SSHafe system provides a tool to enroll Passkeys for the account. The

tool presents a magic link on the SSH terminal itself to enroll Passkeys with Resident Credentials for the user account. This magic link is tightly scoped to expire in 10 minutes and allows enrollment only for the account it is issued for. This passkey is later used for account recovery. The enrollment is done in-band to prevent attacks on email or other transport mechanism. This magic link is single-use with 10 minute expiry, and the security of it is beyond the scope of this study. Algorithm 1 represents the password rotation flow and figure 9 shows the steps of SSHafe Password rotation flow.

Algorithm 1 SSHafe: Password Rotation Algorithm

1: RP Server initiates an Ephemeral ECDH key pair $(\mathrm{sk}_{rp}, \mathrm{pk}_{rp})$
2: pk_{rp} is encapsulated as the `challenge` in a `PublicKeyCredentialRequestOptions`; `allowCredentials` is left empty (usernameless flow)
3: `PublicKeyCredentialRequestOptions` is transmitted to the client
4: Client browser generates its own ephemeral ECDH key pair $(\mathrm{sk}_{br}, \mathrm{pk}_{br})$
5: Browser extracts pk_{rp} from the challenge and computes shared secret $S \leftarrow \mathrm{ECDH}(\mathrm{sk}_{br}, \mathrm{pk}_{rp})$
6: Derive encryption key $K \leftarrow \mathrm{HKDF}(S,\ \mathrm{salt} = \mathrm{pk}_{rp})$
7: Obtain password P from user; sample random IV; compute $C \leftarrow \text{AES-GCM-256}_K(P,\ \mathrm{IV})$
8: Replace `challenge` in `PublicKeyCredentialRequestOptions` with the concatenation $\mathrm{pk}_{br} \| \mathrm{IV} \| C$
9: Complete the WebAuthn assertion flow using a Discoverable (Resident) Credential with the modified `PublicKeyCredentialRequestOptions`
10: Assertion $\mathcal{A}$ is returned to the RP Server
11: RP Server extracts the modified challenge from `clientDataJSON` $\in \mathcal{A}$
12: RP Server extracts `userHandle` from $\mathcal{A}$ and retrieves the corresponding registered public key pk_{user}
13: RP Server verifies assertion signature over modified challenge using pk_{user}
14: **if** verification fails **then**
15: **abort**
16: **end if**
17: RP Server parses the modified challenge to recover pk_{br}, IV, and C
18: Compute shared secret $S \leftarrow \mathrm{ECDH}(\mathrm{sk}_{rp}, \mathrm{pk}_{br})$
19: Derive decryption key $K \leftarrow \mathrm{HKDF}(S,\ \mathrm{salt} = \mathrm{pk}_{rp})$
20: Decrypt $P \leftarrow \text{AES-GCM-256}_K^{-1}(C,\ \mathrm{IV})$; AEAD tag validation defeats replay attacks
21: **if** AEAD tag validation fails **then**
22: **abort**
23: **end if**
24: Set the password for `userHandle` to P

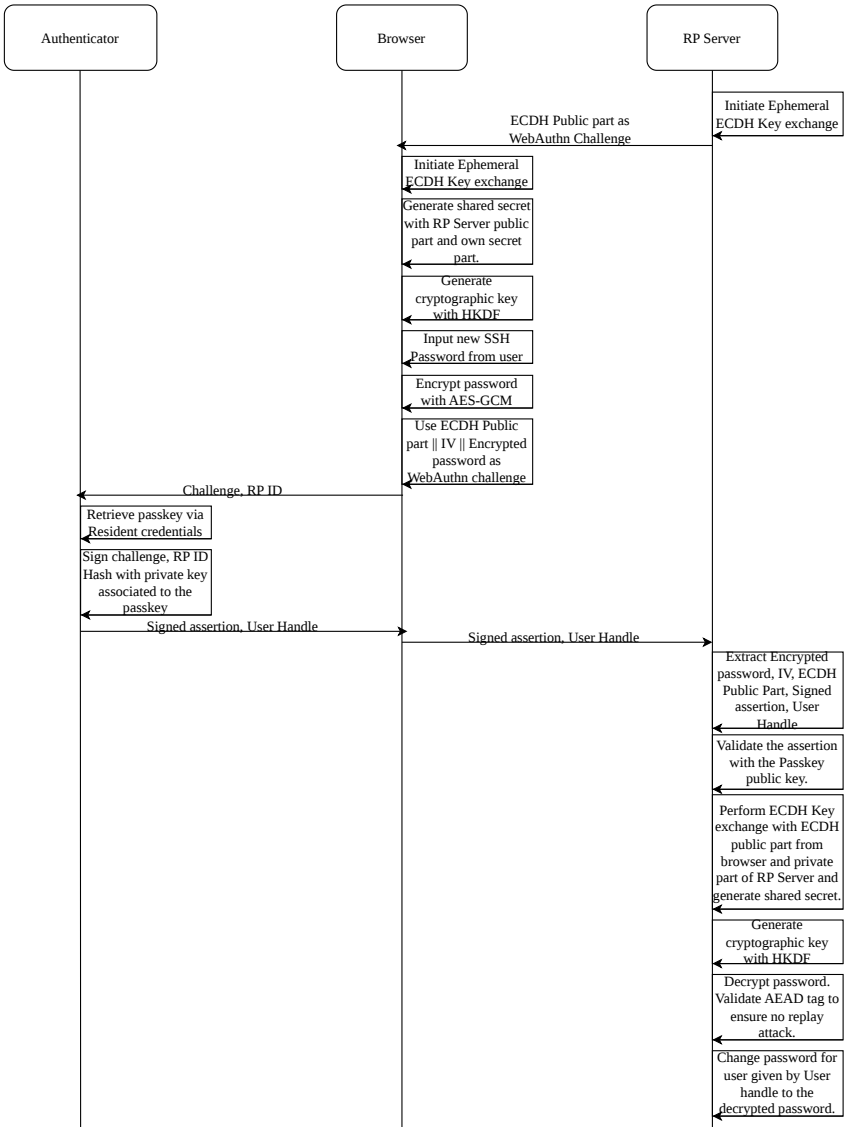


Fig. 9 Password Rotation Flow

4 Experimental Setup and Validation

The LightGBM-based machine learning model for this study was trained on a Google Colab notebook, and the model weights were then used on a cloud VM for real-world validation. An Azure B2ats series Virtual Machine with 2 vCPUs and 1 GB of memory with Ubuntu 24.04 LTS operating system was used. This configuration has nearly the least system capabilities in modern days, which further assures that the performance of the model on machines with better configurations will definitely be better than this VM.

On the other hand, a standard laptop with Kali Linux was used to simulate the attacker. The Hydra tool was used with the Rockyou wordlist, a list of approximately 14.3 million unique passwords. The target VM was able to detect the attacker under 10 seconds in all of 15 tries and disabled the user account. The user account deliberately had a weak password for the tests, yet it could not be taken over by attackers. Figure 10 shows Hydra being used to perform an SSH brute force attack.

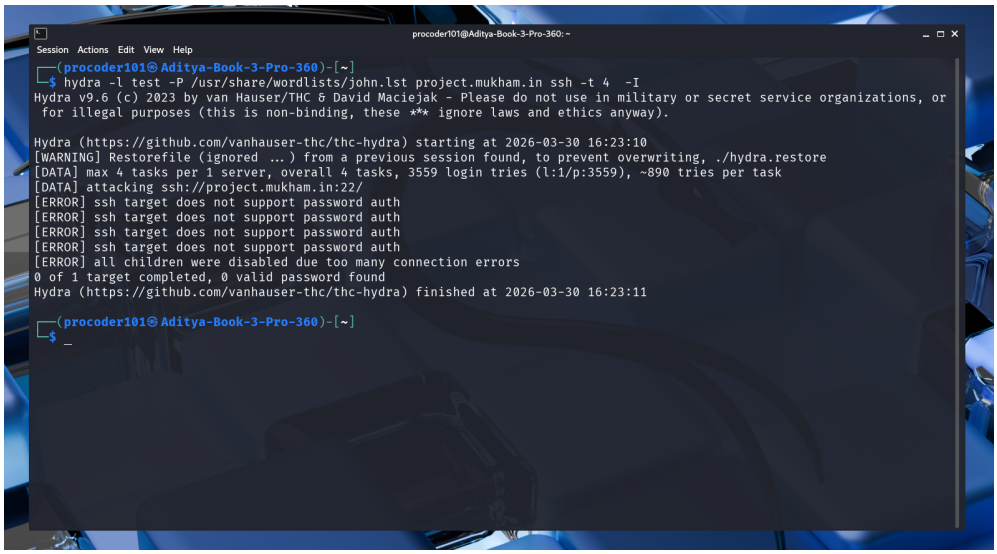


Fig. 10 Hydra attack

This VM was left running on the cloud for two weeks, where it was naturally discovered by real adversarial actors, and they tried to attack the system. The system successfully defended against the attacks from the real adversarial actors, even when both the username and password were relatively weak.

The blocked user accounts showed a banner with a link and QR code to the account recovery portal for legitimate users to be able to recover the accounts. Figure 11 shows the banner used for blocked accounts.

A web-based portal was created for the password reset flow that used the proposed standard to rotate passwords of blocked users. It offered the user to enter the new password, while the username was chosen from the resident credentials of a passkey.

Fig. 11 Account blocked message

Figures 12 shows the homepage while 13 and 14 show the prompts for password and passkey, respectively.

5 Results

The system was able to detect and mitigate brute force attacks on SSH. A control test was done without the proposed system enabled on the same Virtual machine with Hydra. The attack persisted and showed all characteristics of a brute force attack, like high network usage, high disk write for logs, and high CPU usage. Figure 15 shows the resource consumption of the VM under the brute force attack.

Further, an identical test was performed with the same VM but with the proposed system running. It showed a slight spike in inbound network usage, followed by high disk write, showing the existence of a brute force attack. But it was short-lived, as the system detected it and blocked the user account, which later forced Hydra to exit because password authentication was no longer available. The system had a higher outbound network usage than inbound, unlike the control test, and this can be attributed to the system using a large banner for the blocked accounts, which was sent for all blocked brute-force attempts. The CPU did not spike, unlike the control

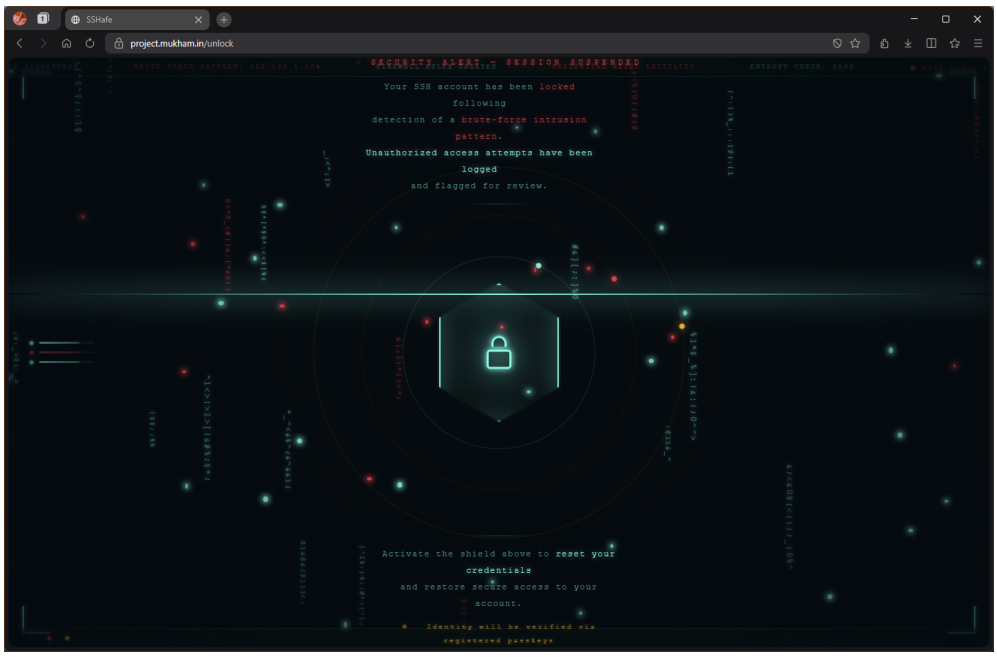


Fig. 12 Password rotation homepage

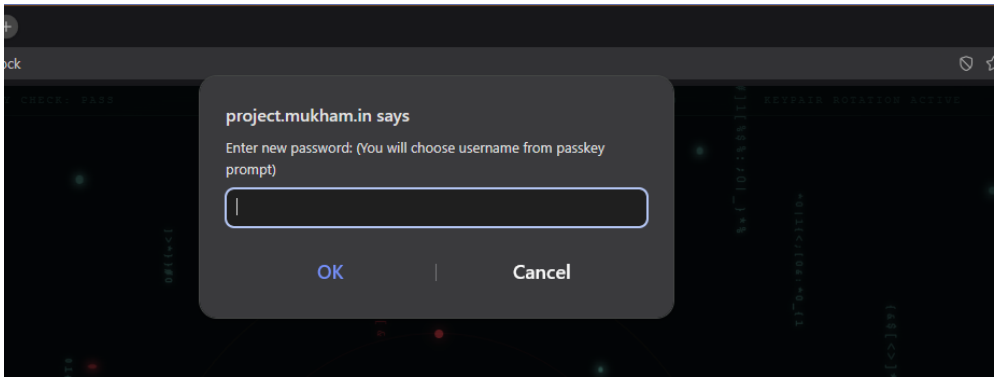


Fig. 13 Prompt for new password

test, and it mitigated the incoming attack. Figure 16 shows the resource usage of the VM on mitigation with the proposed system.

6 Threat model

The proposed system aims to be more effective than traditional SSH attack detection systems like Fail2Ban and provide a more robust system for account recovery. The threat model of the system includes:

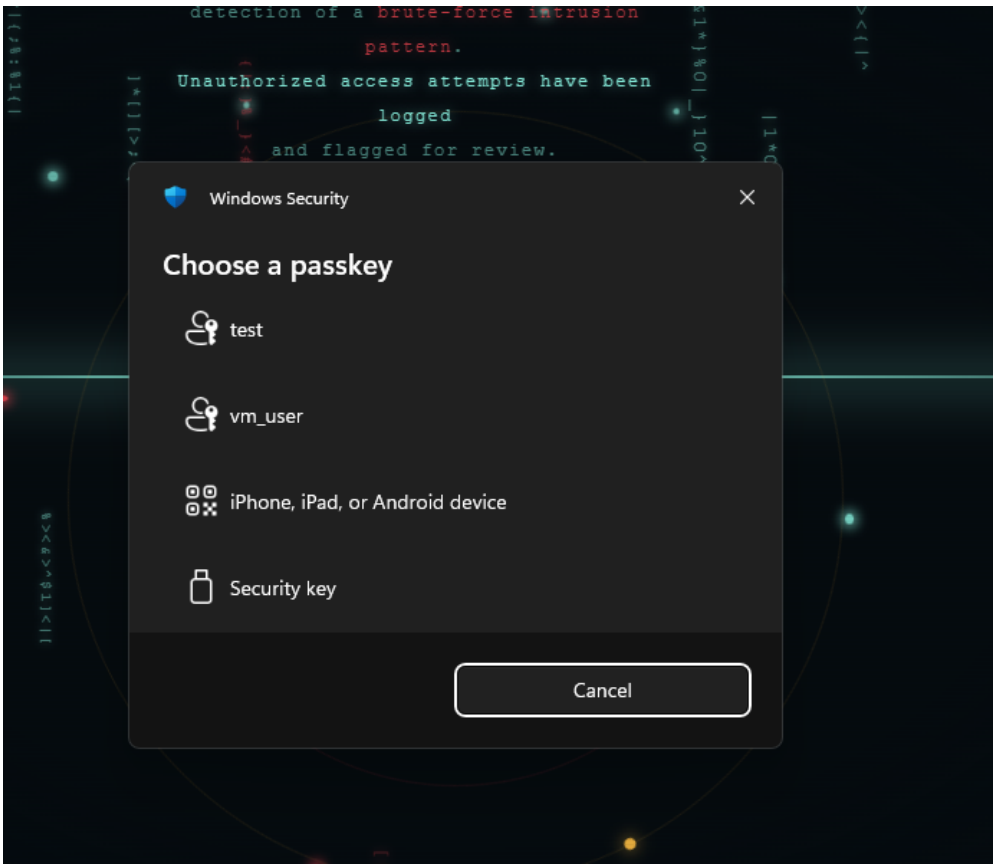


Fig. 14 Passkey choice

- Brute force attacks: The system is aimed to detect and mitigate brute force attacks on the SSH system.
- Dictionary attacks: Dictionary attacks are when an attacker attempts to build a personalized or non-personalized wordlist per user and uses multiple passwords from there. The system is able to mitigate such attacks.
- Password spray: Password spray attacks involve using a number of known passwords till one succeeds. The system is able to mitigate against such attacks.
- Phishing on account recovery flow: Account recovery flow uses a phishing-resistant authentication system and mitigates phishing.
- Unexpired sessions: The account recovery flow does not use long-lived sessions, which can be used by attackers.
- Cookie theft and session hijack: Cookies and Session cookies used in the account recovery flow cannot be used to authenticate the user or the attacker. Hence, any attack involving them would not result in an account takeover.

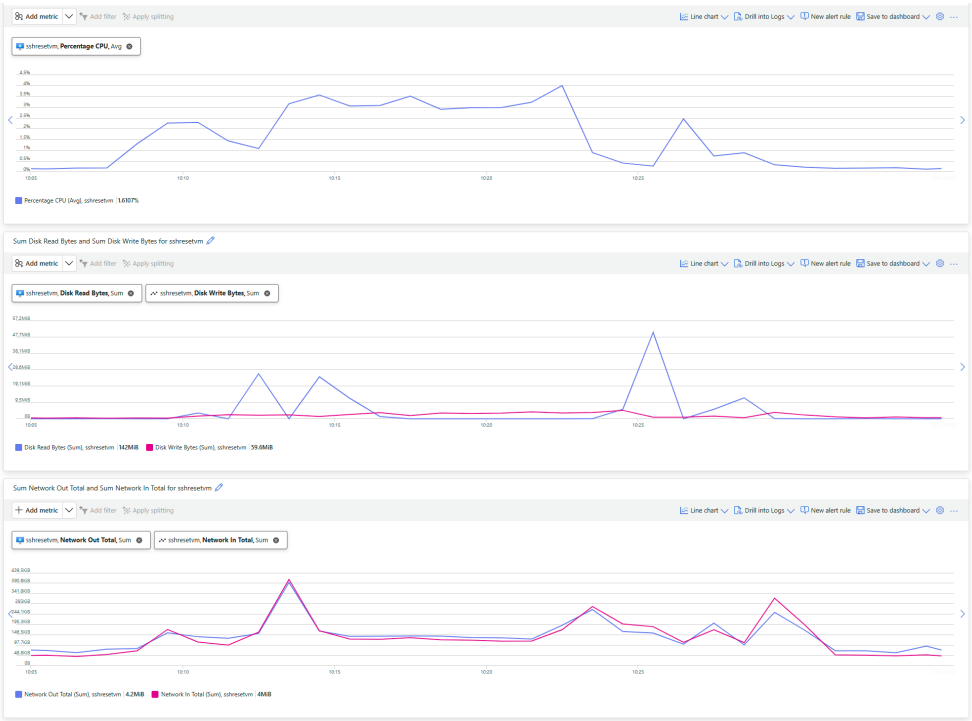


Fig. 15 Stats under attack

- CSRF: The account recovery page does not take inputs regularly and instead derives the username from resident credentials. Hence, it cannot be triggered by CSRF attacks.

7 Conclusion

SSHafe demonstrates to be a solution for fast detection and mitigation of SSH attacks. However, the system may, in future, be combined with other signals from other intrusion detection systems and network and security appliances for better attack detection, not just for SSH, but also for various other types of attacks.

The novel password rotation flow over passkeys may be used for other secret sharing flows as well, not just for account recovery. The password rotation flow aims to be instrumental for sessionless and stateless authenticated secret sharing.

SSHafe has been deployed on real-world cloud environments where it was discovered and attacked by malicious actors and the system has proven to be able to mitigate them, even against real-world adversarial traffic of unknown origin and tooling. The standard proposed in SSHafe would be instrumental in detecting and mitigating various types of attacks in the future, not just SSH.

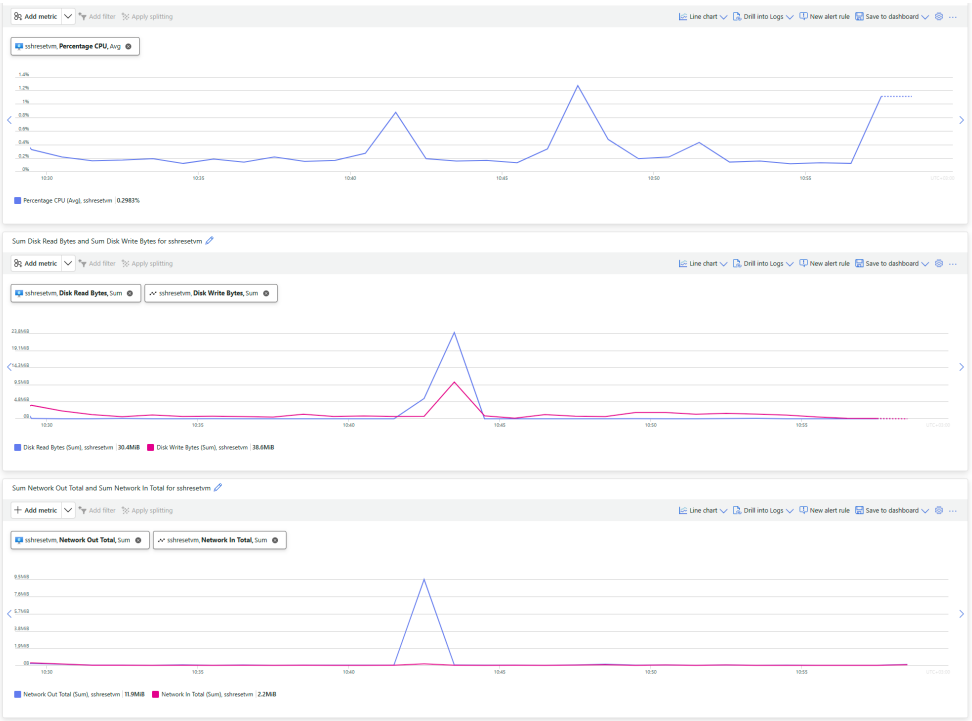


Fig. 16 Stats after mitigation

Acknowledgment

This research was conducted within the framework of the Erasmus Mundus Joint Master's Programme in Applied Cybersecurity (CyberMACS) — Project No. 101082683, co-funded by the European Union under the Erasmus+ MUNDUS Programme.

References

[1] Hynek, K., Beneš, T., Čejka, T., Kubátová, H.: Refined detection of ssh brute-force attackers using machine learning. In: Hölbl, M., Rannenberg, K., Welzer, T. (eds.) ICT Systems Security and Privacy Protection, pp. 49–63. Springer, Cham (2020)

[2] Tiwari, N., Hubballi, N.: Secure socket shell bruteforce attack detection with petri net modeling. IEEE Transactions on Network and Service Management **20**(1), 697–710 (2023) https://doi.org/10.1109/TNSM.2022.3212591

[3] Li, Y., Chen, Z., Wang, H., Sun, K., Jajodia, S.: Understanding account recovery in the wild and its security implications. IEEE Transactions on Dependable and Secure Computing **19**(1), 620–634 (2022) https://doi.org/10.1109/TDSC.2020.2975789

[4] GONZALEZ, S., HERRERO, A., SEDANO, J., ZURUTUZA, U., CORCHADO, E.: Different approaches for the detection of ssh anomalous connections. Logic Journal of IGPL, 047 (2015) https://doi.org/10.1093/jigpal/jzv047

[5] Wanjau, S.K., Wambugu, G.M., Kamau, G.N.: Ssh-brute force attack detection model based on deep learning. International Journal of Computer Applications Technology and Research **10**(01), 42–50 (2021) https://doi.org/10.7753/ijcatr1001.1008

[6] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy (2018). https://api.semanticscholar.org/CorpusID:4707749

[7] Hossain, M.D., Ochiai, H., Doudou, F., Kadobayashi, Y.: Ssh and ftp brute-force attacks detection in computer networks: Lstm and machine learning approaches. In: 2020 5th International Conference on Computer and Communication Systems (ICCCS), pp. 491–497. IEEE, ??? (2020). https://doi.org/10.1109/icccs49078.2020.9118459 . http://dx.doi.org/10.1109/ICCCS49078.2020.9118459

[8] Panigrahi, R.P.: CICIDS2017. IEEE DataPort (2025). https://doi.org/10.21227/AKXQ-9V09 . https://ieee-dataport.org/documents/cicids2017

[9] Sharma, N., Swarnkar, M., Zuhair, M.: Brussh: Early detection of distributed brute force ssh attacks using lstm. In: 2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1–6. IEEE, ??? (2024). https://doi.org/10.1109/ants63515.2024.10898570 . http://dx.doi.org/10.1109/ANTS63515.2024.10898570

[10] Stamatogiannakis, M., Bos, H., Groth, P.: PANDAcap SSH Honeypot Dataset. Zenodo (2020). https://doi.org/10.5281/ZENODO.3759652 . https://zenodo.org/record/3759652

[11] Dananjaya, M.W.P., Candrawengi, N.L.P.I., Suranata, I.W.A., Paramartha, I.G.N.D.: Towards robust ssh anomaly detection: A deep learning comparative analysis. In: 2025 IEEE 2nd International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs), pp. 440–445. IEEE, ??? (2025). https://doi.org/10.1109/icocics68032.2025.11383897 . http://dx.doi.org/10.1109/ICoCICs68032.2025.11383897

[12] Lee, J., Lee, H.: Improving ssh detection model using ipa time and wgan-gp. Computers and Security **116**, 102672 (2022) https://doi.org/10.1016/j.cose.2022.102672

[13] Zare Moayedi, H., Masnadi-Shirazi, M.A.: Arima model for network traffic prediction and anomaly detection. In: 2008 International Symposium on Information Technology, pp. 1–6. IEEE, ??? (2008). https://doi.org/10.1109/itsim.2008.

4631947 . http://dx.doi.org/10.1109/ITSIM.2008.4631947

[14] Wu, Q., Shao, Z.: Network anomaly detection using time series analysis. In: Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services - (icas-isns'05), pp. 42–42. IEEE, ??? (2005). https://doi.org/10.1109/icas-icns.2005.69 . http://dx.doi.org/10.1109/ICAS-ICNS.2005.69

[15] Freeman, R., Bagui, S.S., Bagui, S.C., Mink, D., Cameron, S., Carvalho, G.C.S.D.: A hybrid time series forecasting model combining arima and decision trees to detect attacks in mitre attack labeled zeek log data. Electronics **15**(4), 871 (2026) https://doi.org/10.3390/electronics15040871

[16] Moustafa, N.: The UNSW-NB15 dataset. UNSW Sydney (2019). https://doi.org/10.26190/5D7AC5B1E8485 . http://hdl.handle.net/1959.4/resource/collection/resdatac_445/1

[17] Al-Mazrawe, A., Al-Musawi, B.: Enhancing cloud network security with innovative time series analysis. Journal of Internet Services and Applications **16**(1), 13–24 (2025) https://doi.org/10.5753/jisa.2025.4768

[18] Source, E.D.: NSL-KDD dataset (01/01/2009 to 01/01/2009). IMPACT (2018). https://doi.org/10.23721/100/1478792 . https://www.impactcybertrust.org/dataset_view?idDataset=928

[19] Uwazie, E.C., Iheonkhan, I.S., Adenekan, W.A., Apene, O.Z.: Comparison of convolutional neural networks, long short-term memory networks, and recurrent neural networks for intrusion detection system. In: 2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG), pp. 1–7. IEEE, ??? (2024). https://doi.org/10.1109/seb4sdg60871.2024.10630306 . http://dx.doi.org/10.1109/SEB4SDG60871.2024.10630306

[20] Udurume, M., Shakhov, V., Koo, I.: Comparative analysis of deep convolutional neural network—bidirectional long short-term memory and machine learning methods in intrusion detection systems. Applied Sciences **14**(16), 6967 (2024) https://doi.org/10.3390/app14166967

[21] Zhu, Q., Zhan, X., Chen, W., Li, Y., Ouyang, H., Jiang, T., Shen, Y.: Gatransformer: A network threat detection method based on graph-sequence enhanced transformer. Electronics **14**(19), 3807 (2025) https://doi.org/10.3390/electronics14193807

[22] Nguyen, H., Hajisafi, A., Abdoli, A., Kim, S.H., Shahabi, C.: An evaluation of time-series anomaly detection in computer networks. In: 2023 International Conference on Information Networking (ICOIN), pp. 104–109. IEEE, ??? (2023). https://doi.org/10.1109/icoin56518.2023.10049051 . http://dx.doi.org/10.1109/ICOIN56518.2023.10049051

[23] Ali, M.L., Thakur, K., Schmeelk, S., Debello, J., Dragos, D.: Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study. Applied Sciences **15**(4), 1903 (2025) https://doi.org/10.3390/app15041903

[24] Tiwari, N., Hubballi, N.: Secure socket shell bruteforce attack detection with petri net modeling. IEEE Transactions on Network and Service Management **20**(1), 697–710 (2023) https://doi.org/10.1109/tnsm.2022.3212591

[25] Li, Y., Chen, Z., Wang, H., Sun, K., Jajodia, S.: Understanding account recovery in the wild and its security implications. IEEE Transactions on Dependable and Secure Computing **19**(1), 620–634 (2022) https://doi.org/10.1109/tdsc.2020.2975789

[26] Sun, H.-M., Chen, Y.-H., Lin, Y.-H.: opass: A user authentication protocol resistant to password stealing and password reuse attacks. IEEE Transactions on Information Forensics and Security **7**(2), 651–663 (2012) https://doi.org/10.1109/tifs.2011.2169958

[27] Yusop, M.I.M., Kamarudin, N.H., Suhaimi, N.H.S., Hasan, M.K.: Advancing passwordless authentication: A systematic review of methods, challenges, and future directions for secure user identity. IEEE Access **13**, 13919–13943 (2025) https://doi.org/10.1109/access.2025.3528960

[28] Sudhodanan, A., Paverd, A.: Pre-hijacked accounts: An empirical study of security failures in user account creation on the web. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 1795–1812. USENIX Association, Boston, MA (2022). https://www.usenix.org/conference/usenixsecurity22/presentation/sudhodanan

[29] Innocenti, T., Mirheidari, S.A., Kharraz, A., Crispo, B., Kirda, E.: You've Got (a Reset) Mail: A Security Analysis of Email-Based Password Reset Procedures, pp. 1–20. Springer, ??? (2021). https://doi.org/10.1007/978-3-030-80825-9_1 . http://dx.doi.org/10.1007/978-3-030-80825-9_1

[30] Dananjaya, M.W.P.: SSH Anomaly Dataset. Kaggle. Accessed: March, 2026 (2024)