OPEN ACCESS

Kashf Journal of Multidisciplinary Research
Vol: 02 - Issue 09 (2025)
P-ISSN: 3007-1992 E-ISSN: 3007-200X
https://kjmr.com.pk

# HONEYTOKEN-AUGMENTED AI MODEL FOR INSIDER THREAT DETECTION USING CYBER DECEPTION AND ISOLATION FOREST

***[1]Dr. Ajab Khan ,[2]Usman Imtiaz Member IEEE , *[3]Fahad Amin Member IEEE***

*[1]University of Science and Technology, Abbottabad, Pakistan.*

*[2]Washington University of Science and Technology (WUST), Department of Computer Science-Cyber security, Virginia, USA*

*[3]North American University, Department of Computer Science-Cyber security, Houston, TX, USA*

****Corresponding Author:*** (famin1@na.edu)

***DOI:*** (*https://doi.org/10.71146/kjmr625*)

**Article Info**



**Abstract**

*It is quite hard to detect insider threats as malevolent users may use legitimate credentials and authorized access. Rule-based cyber security systems do not work on insider threats since abnormal behavior does not transgress a predefined rule. AI can do a better job by understanding user's normal behavior and distinguishing anomaly behavior. We proposed Honeytoken-Augmented Deception Isolation Forest model named HAD-IF for the detection of insider threats. It combines behavioral anomaly detection and honeytoken based cyber deception. The anomaly detection model, Isolation Forest, identifies the anomaly based on following parameters of user's activity; (1) Login Time, (2) Amount of file accessed, (3) Number of unsuccessful login attempt, (4) Access to critical files, (5) Amount of data downloaded, and honeytoken based cyber deception. Honeytoken is an attractive digital bait that generates severe alert when interacted by malicious users. A Python implementation of HAD-IF on synthetic user activity logs is developed. Experiments result in terms of Accuracy, Precision, Recall and F1-score are 97.00%, 86.96%, 100.00% and 93.02% on the simulation dataset respectively. The results indicate that the combination of AI-based anomaly detection and cyber deception technology improves early insider threats detection.*

**Keywords:** *Artificial Intelligence, Cyber Security, Insider Threat Detection, Honeytoken, Cyber Deception, Isolation Forest, Machine Learning, Anomaly Detection, Risk Scoring, User Behavior Analytics, Threat Intelligence, Security Monitoring.*

## I. Introduction

Cyber security, also called information technology security, protection of computer systems, networks, application and digital information against unauthorized use, access, deletion, destruction and disruption. The primary goal of cyber security is to provide availability, confidentiality, integrity to the digital assets. The business has increasingly relied upon cloud platforms, online database systems, remote access, application and automated systems for logging activities. This has also widened the range of possible vulnerabilities; allowing attacks to penetrate systems with the use of weak credentials, poor access control, badly configured systems and unnoticed user activity [1].

A common misinterpretation about cyber security, however, is that every attack must come from external perpetrators. In fact, attackers could be employees or contractors who have unauthorized access to systems or devices, or insider personnel or system administrators whose privileges have been compromised, or system users like students. An insider attack is an act by any individual with authorized access to computer system and data, who then misuses their access to violate the security of information [2],[3]. Insider attack could be malicious or unintentional. An intentionally malicious insider may compromise data for profit or revenge through actions like stealing important files, leaking secret documents, misusing administrative privileges or sabotaging system components [2],[3]. An unintentionally malicious insider, otherwise called an incompetent user, would compromise data through lack of knowledge, like selecting weak password, failing to adhere to security policies or clicking on phishing emails and ultimately placing secret files at risk by storing them at easily accessible places [2],[3]. Though rule-based security systems are useful they are not adequate. While they would raise a flag when a user fails five consecutive logins or attempts to access a privileged directory. A determined insider would try to evade this, possibly trying smaller number of files or operating at unconventional hours, thereby spreading their activities throughout various login sessions. Thus, sophisticated modern system security must learn patterns and detect abnormal human behavior [4]. Artificial Intelligence, especially Machine Learning (ML) can aid cyber security systems to detect anomalies. Through analysis of the log files. Rather than simply checking rules, AI models may identify certain unusual combinations of factors like login after normal working hours, a massive jump in file accessed, several unsuccessful login attempts, an access of a great number of secret files and a huge download volume. A certain factor in the list would not look that much suspicious alone but all put together will define an abnormal behavior, thereby useful for anomaly detection [4],[5].

This paper suggests an AI prototype named HAD-IF for defense using deception based methods by combining anomaly detection and honey-tokens. Anomaly detection could be realized with an Isolation Forest model in which normal human behavior is identified by the activity features, and abnormal human activity receives an anomaly score. Isolation Forest works best at quickly finding anomalous data because it isolates abnormal points from normal points more effectively [5]. For that reason, it performs well in detecting low-frequency abnormal user actions in log-based systems.

Additionally, a second layer of deception-based detection is introduced with the use of honeytokens. Honeytokens are simulated files, fake accounts, fake records and fake documents which has no business relevance. When they are accessed by anyone it is taken as a clear alert, because regular business personnel will always avoid touching anything that looks fake or are of no business relevance [6],[7].

So, this HAD-IF combines these two signals of anomaly detection using ML and the signals from the honeytoken. For example, a user logging in at unusual hours and downloading files that looks suspicious will get high anomaly score. With the same user interacting with the honeytoken file, the severity can be escalated to "critical", because that shows malicious insider intent. This enhances the detection of threats as it depends on not one but on combined signals of two independent security methods.

**This paper makes the following contributions:**

An AI based cyber deception approach for insider-threat risk assessment.
A condensed feature set for evaluating a user's risk score through examination of login times, the number of files accessed, failed logins, number of secret files accessed and the download data volume.
A honeytoken signal identifies abnormal user actions and escalates them to "critical" risk level whenever a false asset is accessed. A basic prototype of a defensive AI system written in python with charts, a model architecture diagram, algorithm designs, and experimental findings.
The objective of this research paper is entirely based on defense, education, and demonstration of how to combat malicious insider actions utilizing Artificial Intelligence and cyber deception. Our goal is not to engage in hacking, steal data, or compromise system defenses. By implementing this research in a practical, well-considered and thought-provoking way, organizations will be able to detect suspicious insider behavior more promptly and more accurately. However, there are numerous privacy and fairness issues associated with insider-threat detection systems that need to be carefully addressed. NIST AI Risk Management Framework, for example, specifies that an AI system must be valid, secure, reliable, explainable, privacy-preserving and responsible [8]. Insider-threat detection systems must be put into service with specific organizational guidelines, user permissions, a qualified human security analyst to examine all suspicious actions, and strict security measures to protect data.

## II. Background

### Artificial Intelligence in Cyber Security:

AI is used for threat detection, malware analysis, phishing message classification, network traffic monitoring, fraud detection and incident response in cyber security. Due to large volumes of logs, alerts and events within a Security Operations Center, it is very helpful to apply machine learning models, because those are able to perform the work more quickly than human analysts. In SOC, AI could assist analysts by prioritizing alerts, to make them spend more time on high-risk activities and less on noise-generating alerts. It also helps to intrusion detect using abnormal patterns in network traffic, user actions, and endpoint events. AI should not replace human analysts but used as a decision support tool for quick detection of high-risk events and reduce alert fatigue [9].

**B. Insider Threats:**

Insider threats differ from the many other external threats, in that, an insider has some kind of trust within the organization. It can be seen in the way he/she interacts with the systems. One may open a confidential file (if for instance their role dictates that they must) and on the other hand a regular office worker accessing the same files would be suspicious. It is also possible to observe how some user uploads a large amount of data (which for a data analyst may be normal behavior but for an office worker suspicious). Therefore, it is difficult to differentiate between innocent behavior and malicious activity, which requires context-based detections using baseline behaviors and risk scoring rather than yes/no evaluations. This includes malicious employees, careless employees, compromised accounts, contractors, administrators or other privileged users, who access any part of the system with or without their intent [10], [11]. Due to insider ability to bypass most of the perimeter defenses it is necessary to introduce monitoring mechanisms based on the user and role, previous activity patterns, authentication behavior, and access level to certain data [12].

**C. Anomaly Detection:**

Anomaly detection is the identification of objects and events that deviate from what is considered expected. Within cyber security these may include intrusions, compromised accounts, data exfiltration, stolen credentials, privilege abuse, or violations of defined security policies. Anomaly detection algorithms are beneficial for insider threat detection as insider misuse is expected to be rare when compared with the volume of normal user activity; which may result in an imbalanced dataset and a need for unsupervised learning in the absence of a large set of labeled examples of an attack [13]. The isolation forest algorithm could be an interesting alternative for such task due to its ability to isolate anomalies more quickly than observations representing normal behavior. Using HAD-IF model, this kind of anomaly detection may serve to provide risk scores for user sessions and to be based on features, such as time of log in, number of failed log in attempts, accessing sensitive data, download data volume, and more [14].

**D. Cyber Deception:**

Cyber deception is a proactive defense mechanism to bait and discover malicious activities by luring attackers with false assets, misleading signals, or baiting setups. Different from conventional security tools that work by blocking and filtering information flow, the tools used in deception technologies enable defenders to observe malicious behavior and to learn about attacker intentions. Examples are honeypots, honeyfiles, honeypot accounts, honeypot API keys, honey data rows, deception folders, honeytokens, etc. These dummy resources aim to represent plausible but normally inaccessible real resources. Hence, the interaction of the actor with such assets could be strongly indicative of him performing reconnaissance, unauthorized data access or any sort of intrusion. Engaging into adversary engagement on the other hand represents an attempt by defenders to manipulate attackers with planned

deceptive tactics and denial and thus improve the detection ability of threat [15]. Early studies about honeypots have even demonstrated how to discover, identify and profile malicious insider behavior [16]. Honeytokens-pretend but realistic digital objects-are deployed as decoys. Such an object may be a document titled admin_passwords.txt, a fake database record, a dummy username/password, a fake cloud access key, or a decoy file within a location that seems sensitive. It appears valuable and useful but carries no actual business value. The object is, in effect, a tripwire when accessed, generating an alert. The value of honeytokens lies in the fact they don't just prevent malicious access, but they also reveal curiosities, reconnaissance activities, unauthorized access attempts, or other malicious intent. For instance, when an ordinary employee visits a fake payroll database or accesses fake administrator credentials, this can be perceived as a critical alert. Indeed, research in decoy documents revealed that fake documents could also lure and identify internal threats based on their interaction with unauthorized documents [17]. Thus, in the HAD-IF framework, access to honeytokens is deemed a high risk signal. Honeytokens are raised to a critical risk signal if they co-occur with an abnormal activity recognized by the AI model.


Figure 1 Operational risk-scoring workflow for AI-assisted cyber deception.

## III. Related Work

NIST's AI Risk Management Framework guides the trustworthy and responsible management of AI risk. In the context of cybersecurity research, AI models should be evaluated, monitored, documented, and governed rather than blindly implemented. A poorly chosen training set or changes in the operating environment may lead to a false positive, false negative, biased result or unstable output from an AI-based security model, necessitating testing before deployment, monitoring post-deployment and review by human security analysts. NIST AI RMF principles such as validity, reliability, safety, security,

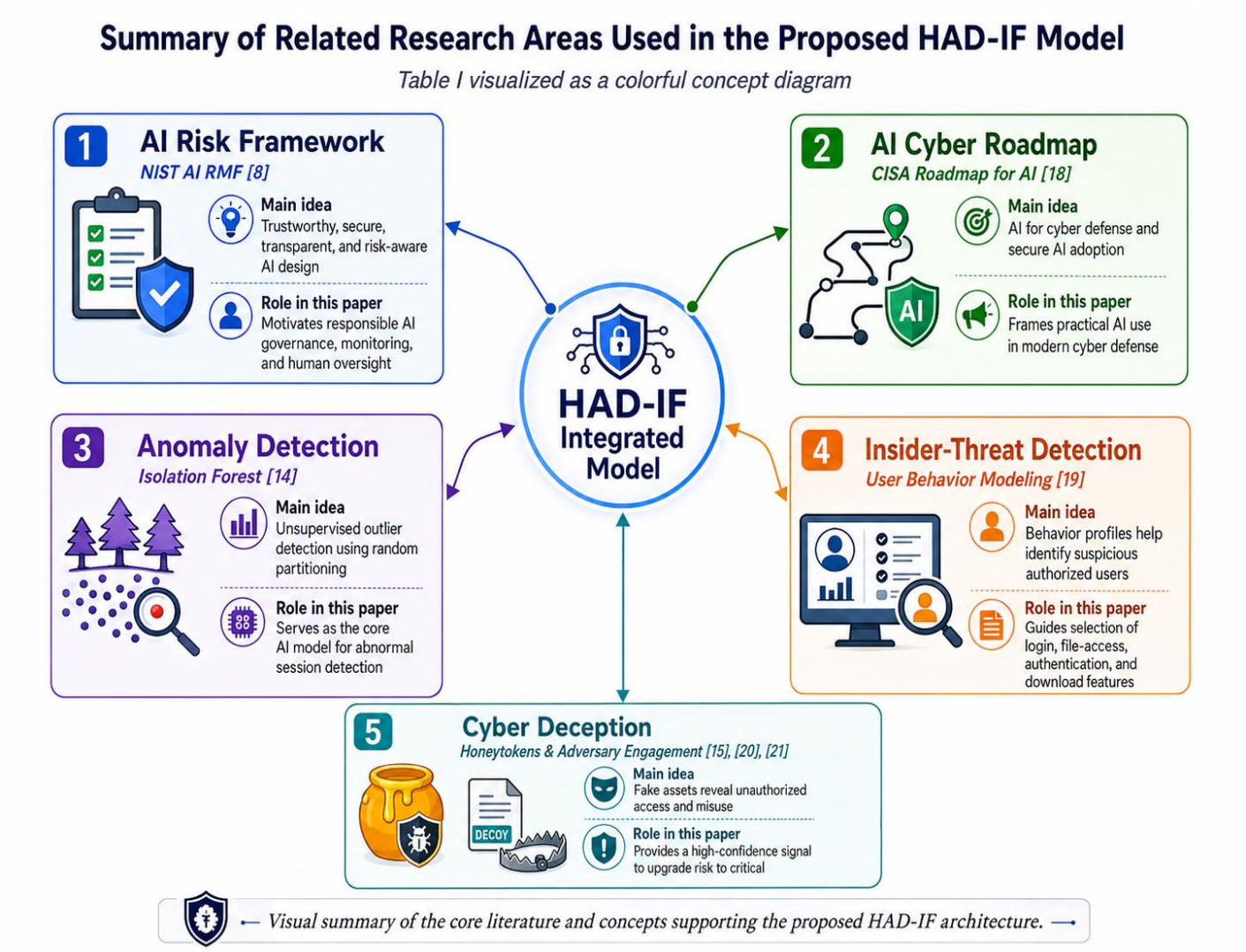
resilience, accountability, transparency, explainability, privacy enhancement, and fairness are essential in an insider-threat context, where user activity can be highly sensitive [8]. CISA's Roadmap for Artificial Intelligence similarly identifies AI's potential for improved cybersecurity defense while also outlining the importance of securing AI systems against misuse and attack. AI can both advance cybersecurity and enhance defenders' ability to respond to modern threats. It is also critical that these systems be protected against attack and misuse, which in the case of the work here requires that the proposed HAD-IF model detect unusual insider behavior, while also ensuring security, accountability, and proper governance [18].

Isolation Forest was introduced as an anomaly detection algorithm by Liu, Ting, and Zhou. In this approach, anomalies are isolated by randomly partitioning instances, based on the theory that anomalies are rare and therefore require fewer partitions to separate them from normal instances. This characteristic can be applied in security log analysis to flag user sessions that behave abnormally compared to typical usage patterns; for instance, a user login followed by an unusual number of failed login attempts, a spike in sensitive-file access and excessive downloads, and a login time outside the user's usual hours would likely be flagged quickly by the model [14]. In their work on insider-threat detection using behavior modeling and anomaly detection algorithms, Kim et al. Demonstrated the utility of building user-specific behavior profiles from log data rather than relying solely on predefined rules, as what constitutes "normal" behavior varies from user to user, whether they are a system administrator, finance officer, student, contractor, or regular employee. Kim et al. Also noted that the datasets used in insider-threat detection tend to be imbalanced, with a preponderance of normal activity versus unusual or malicious insider activity [19].

Akiyama et al.'s HoneyCirculator investigated web-based attack cycles using credential honeytokens, distributing bait credentials to trace adversary behavior in interacting with decoy services. This work is relevant to the present study in that it demonstrates the value of observing suspicious behavior through honeytokens rather than simply blocking it. A honeytoken essentially acts as a detectable signal: the compromise of a honeytoken provides strong evidence of an intruder's interaction with an asset without legitimate purpose [20]. Recent work by Prabhaker et al. Extended the use of honeytokens into relational databases to deceive attackers. This approach is particularly relevant in an enterprise environment where much of a company's sensitive data is stored in databases. The use of honeytokens at the database level provides a means of detecting data exfiltration, unauthorized access and suspicious queries. In the context of the proposed HAD-IF model, the idea of using fake sensitive assets as an additional detection signal supports this approach. By combining the detection signals from a honeytoken interaction with the results from an anomaly detection algorithm, the system can assign a greater risk score to suspect sessions and focus analyst attention on truly serious incidents [21].

Anomaly detection and honeytokens have both been explored extensively in cyber security. Anomaly detection algorithms, like Isolation Forest, have proven effective in identifying statistically unusual behavior. Honeytokens are an excellent means of eliciting active responses from intruders that clearly

demonstrate suspicious or malicious intent. The novelty in this paper is the simple, testable design of an integrated prototype combining these approaches with risk-level classification into a single defensive workflow. The proposed work is not an attempt to present an entirely novel concept but rather an exercise in applied education, combining well-established techniques into a clear and readily applicable demonstration of insider-threat detection.




## IV. Proposed HAD-IF Model

The proposed model is named HAD-IF, where HAD-IF stands for Honeytoken-Augmented Deception Isolation Forest. HAD-IF is a defensive insider-threat detection mechanism. It does not attack any system, neither does it steal any password or genuine sensitive data. It processes the simulated or allowed user activity log and classifies user sessions into different risk levels ranging from low to critical-low, medium, high, and critical.

The model comprises five core layers-the data collection layer, feature extraction layer, preprocessing layer, Isolation Forest layer, deception layer, and risk-scoring layer. Data collection layer is responsible for taking in user activity log from login log, file access log, authentication log and downloads log. Feature extraction layer processes the raw log to extract numerical features. Preprocessing layer is responsible for scaling these features so that large range of numbers doesn't overshadow the other. Isolation Forest layer processes the sessions to identify anomalies. Deception layer checks if the user has accessed a honeytoken. Risk-scoring layer combines AI result with deception information.

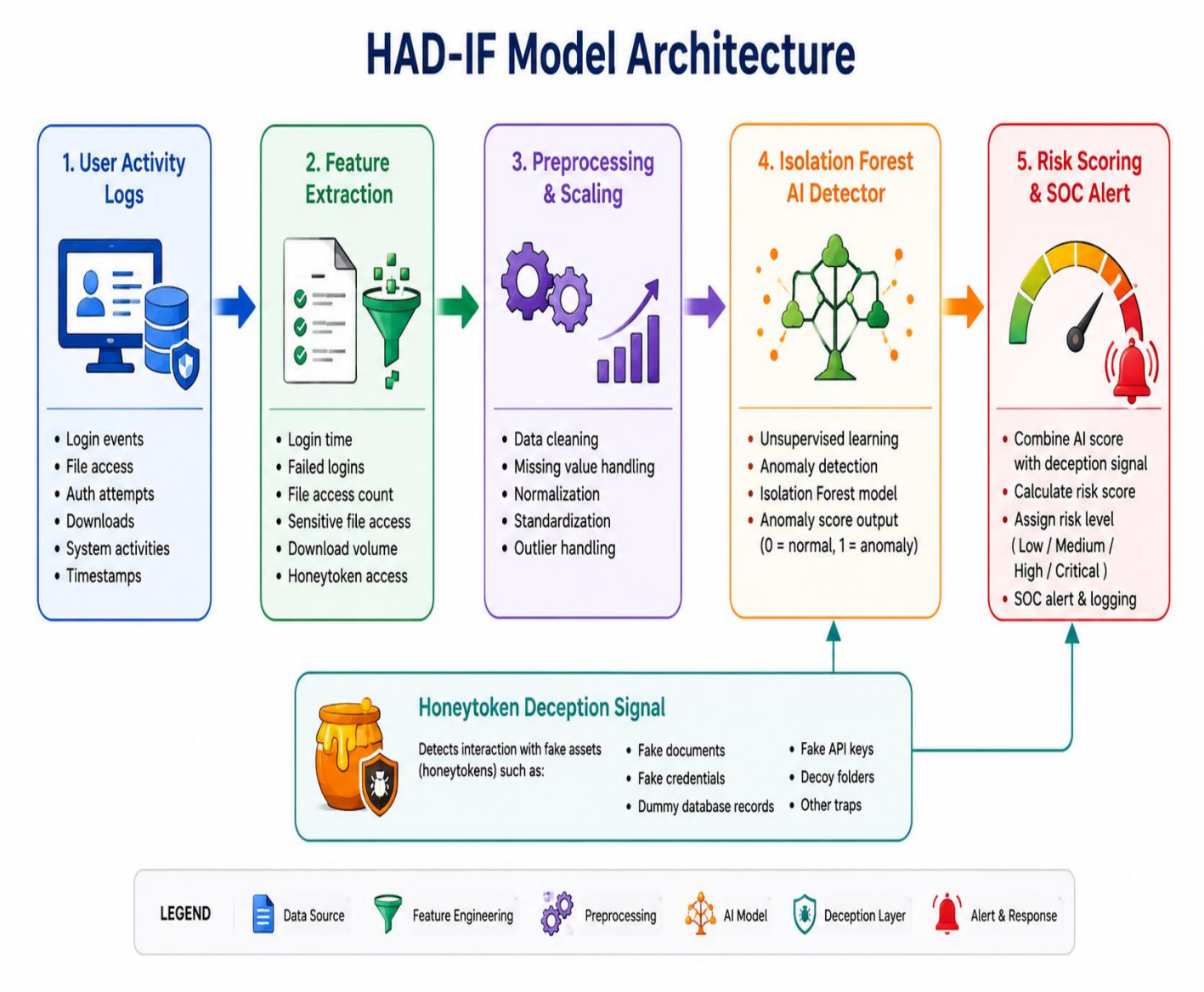



Figure 2 Proposed HAD-IF model architecture.

The model uses the following behavioral and deception features.

Table 1 Feature set and security interpretation.

| Feature | Type | Meaning | Security Interpretation |
|---|---|---|---|
| **login_hour** | Numerical | Hour of login session | Unusual late-night or early-morning access may be suspicious |
| **files_accessed** | Numerical | Number of files opened | A sudden increase may indicate reconnaissance or data theft |
| **failed_logins** | Numerical | Failed authentication attempts | Repeated failures may indicate credential guessing or misuse |
| **sensitive_files** | Numerical | Confidential files accessed | High access to sensitive resources increases risk |
| **download_mb** | Numerical | Volume of downloaded data | Large downloads may indicate exfiltration |
| **honeytoken_access** | Binary | Whether a fake asset was accessed | Triggers critical risk because normal users should not use traps |

The final prediction rule is defined as follows. Let A_i be the anomaly flag produced by Isolation Forest for session i, and let H_i be the honeytoken access flag. The predicted suspicious label is 1 when either the AI detector marks the session as anomalous or the honeytoken signal is activated:

$$\hat{y}_i = 1 \text{ if } (A_i = 1) \text{ OR } (H_i = 1), \text{ otherwise } \hat{y}_i = 0.$$

This rule is intentionally simple because the objective of the prototype is explainability. A security analyst can clearly see whether the alert came from behavioral anomaly, honeytoken interaction, or both.

## V. Risk Scoring and Algorithm

Risk scoring is necessary because not every suspicious action has the same severity. A single unusual login may require review, but opening a fake credential file should be treated as a stronger indicator. HAD-IF therefore assigns four risk levels.

Table 2 Risk-level classification rules.

| Risk Level | Condition | Recommended Action |
|---|---|---|
| Low | No anomaly and no honeytoken access | No immediate alert |
| Medium | Anomaly detected but sensitive access is below threshold | Review if repeated |
| High | Anomaly detected with high sensitive-file access | Security analyst investigation |
| Critical | Honeytoken accessed | Immediate incident-response alert |

**Algorithm 1:** HAD-IF Insider Threat Detection
**Input:** User activity logs L
**Output:** Risk level and alert status for each session

**1.** Start
**2.** Collect authorized user activity logs from login, file, and authentication systems.
**3.** Extract features: login_hour, files_accessed, failed_logins, sensitive_files, download_mb.
**4.** Normalize numerical features using StandardScaler.
**5.** Train Isolation Forest on normal user sessions.
**6.** For every new session i, predict anomaly flag A_i.
**7.** Read honeytoken flag H_i from the deception monitoring layer.
**8.** If H_i = 1, assign Critical risk.
**9.** Else if A_i = 1 and sensitive_files >= threshold, assign High risk.
**10.** Else if A_i = 1, assign Medium risk.
**11.** Else assign Low risk.
**12.** Store risk result and generate alert according to policy.
**13.** End.

The algorithm is designed for a defensive monitoring environment. It assumes that user monitoring is authorized by organizational policy and that honeytokens are created as fake assets under the control of the defender. The model should not be used for secret surveillance outside legal and ethical rules.

## VI. Python Implementation

The following Python code implements the complete prototype using NumPy, pandas, scikit-learn, and matplotlib. The dataset is synthetic, which means it is generated for demonstration. The same logic can be applied to real logs after proper authorization, anonymization, and preprocessing.

```
import numpy as np
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, classification_report

np.random.seed(42)
n_normal, n_obvious, n_stealth = 80, 15, 5

normal = pd.DataFrame({
    "login_hour": np.random.normal(11, 2, n_normal).clip(8, 18).round().astype(int),
    "files_accessed": np.random.normal(18, 7, n_normal).clip(3, 45).round().astype(int),
    "failed_logins": np.random.poisson(0.4, n_normal).clip(0, 3),
```

```
    "sensitive_files": np.random.poisson(2, n_normal).clip(0, 8),
    "download_mb": np.random.normal(65, 25, n_normal).clip(5, 180).round(1),
    "honeytoken_access": np.zeros(n_normal, dtype=int),
    "true_label": np.zeros(n_normal, dtype=int)
})

obvious = pd.DataFrame({
    "login_hour": np.random.choice([0,1,2,3,5,6,22,23], n_obvious),
    "files_accessed": np.random.normal(70, 25, n_obvious).clip(20, 150).round().astype(int),
    "failed_logins": np.random.poisson(4, n_obvious).clip(0, 12),
    "sensitive_files": np.random.normal(20, 8, n_obvious).clip(5, 45).round().astype(int),
    "download_mb": np.random.normal(550, 180, n_obvious).clip(120, 1000).round(1),
    "honeytoken_access": np.random.binomial(1, 0.35, n_obvious),
    "true_label": np.ones(n_obvious, dtype=int)
})

stealth = pd.DataFrame({
    "login_hour": np.random.normal(11, 1, n_stealth).clip(9, 14).round().astype(int),
    "files_accessed": np.random.normal(16, 4, n_stealth).clip(8, 28).round().astype(int),
    "failed_logins": np.random.poisson(0.2, n_stealth).clip(0, 2),
    "sensitive_files": np.random.poisson(2, n_stealth).clip(0, 6),
    "download_mb": np.random.normal(70, 20, n_stealth).clip(20, 130).round(1),
    "honeytoken_access": np.ones(n_stealth, dtype=int),
    "true_label": np.ones(n_stealth, dtype=int)
})

df = pd.concat([normal, obvious, stealth], ignore_index=True)
features = ["login_hour", "files_accessed", "failed_logins", "sensitive_files", "download_mb"]
train_normal = df[df["true_label"] == 0].sample(60, random_state=1)

scaler = StandardScaler()
X_train = scaler.fit_transform(train_normal[features])
X_all = scaler.transform(df[features])

model = IsolationForest(n_estimators=150, contamination=0.05, random_state=7)
model.fit(X_train)

df["iforest_flag"] = (model.predict(X_all) == -1).astype(int)
df["had_if_pred"] = ((df["iforest_flag"] == 1) | (df["honeytoken_access"] == 1)).astype(int)
```

```
def assign_risk(row):
    if row["honeytoken_access"] == 1:
        return "Critical"
    elif row["iforest_flag"] == 1 and row["sensitive_files"] >= 15:
        return "High"
    elif row["iforest_flag"] == 1:
        return "Medium"
    else:
        return "Low"

df["risk_level"] = df.apply(assign_risk, axis=1)

print("Accuracy:", accuracy_score(df["true_label"], df["had_if_pred"]))
print("Precision:", precision_score(df["true_label"], df["had_if_pred"]))
print("Recall:", recall_score(df["true_label"], df["had_if_pred"]))
print("F1-score:", f1_score(df["true_label"], df["had_if_pred"]))
print("Confusion matrix:\n", confusion_matrix(df["true_label"], df["had_if_pred"]))
print(classification_report(df["true_label"], df["had_if_pred"]))
print(df["risk_level"].value_counts())
```

Sample output from the implementation is shown below. The output confirms that the complete HAD-IF design detects all suspicious sessions in the simulation while producing three false positives.

```
Accuracy: 0.97
Precision: 0.8696
Recall: 1.0000
F1-score: 0.9302
Confusion matrix:
[[77  3]
 [ 0 20]]
Risk levels: Low=77, Medium=6, High=7, Critical=10
```

## VII. Experimental Setup

The experiment uses an artificial log file which includes 100 sessions. 80 sessions are normal, 20 sessions are malicious. The normal sessions are produced with the context of a common office behavior: they include a login within the daytime, low number of failed logins, low number of accessed files containing sensitive information, a medium amount of downloads. The malicious sessions have been designed with 15 obvious suspicious sessions which include abnormal behavioral activity, and 5 stealth suspicious sessions which present normal behavior in terms of activity, but do access the honeytoken. It

thus seems possible to conduct an ablation study on this issue, since IF alone may not be able to detect stealth attacks whereas the HAD-IF may still detect it with evidence from the deception feature.

Only normal logs have been fed to the Isolation Forest model. This is because we commonly consider that an organization usually has a good amount of normal activity but fewer proven cases of insider threat. By using normal log records only, the Isolation Forest will learn a normal behavioral signature and will then classify anomalies as abnormal ones. Two configurations of the experiment have been tested. The first uses only the Isolation Forest while the second uses the whole design of HAD-IF with both honeytoken access and the AI flag for abnormality detection. It will thus be possible to evaluate the gain in terms of detection provided by the deception feature.

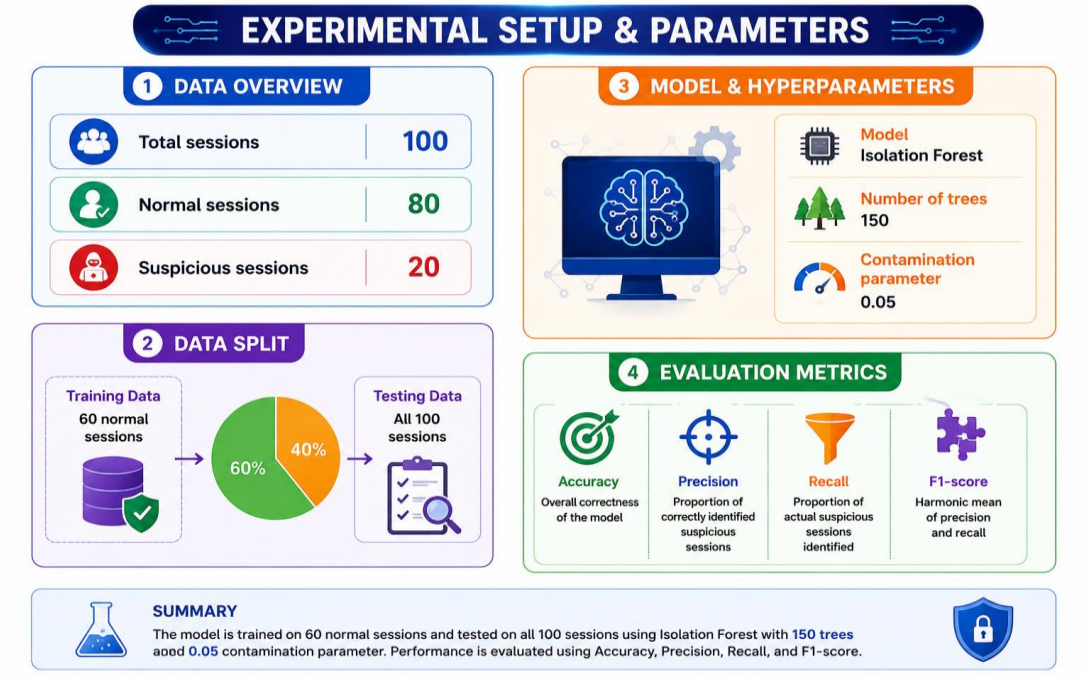



**Table 3 Experimental configuration.**

| Parameter | Value |
|---|---|
| Total sessions | 100 |
| Normal sessions | 80 |
| Suspicious sessions | 20 |
| Training data | 60 normal sessions |
| Testing data | All 100 sessions |
| Model | Isolation Forest |
| Number of trees | 150 |
| Contamination parameter | 0.05 |
| Evaluation metrics | Accuracy, Precision, Recall, F1-score |

## VIII. Results

The complete HAD-IF model achieved the following performance on the simulated dataset.

**Table 4 HAD-IF model performance.**

| Metric | Result |
|---|---|
| Accuracy | 97.00% |
| Precision | 86.96% |
| Recall | 100.00% |
| F1-score | 93.02% |

**Table 5 Confusion matrix of HAD-IF.**

| | Predicted Normal | Predicted Suspicious |
|---|---|---|
| Actual Normal | 77 | 3 |
| Actual Suspicious | 0 | 20 |

The confusion matrix shows 77 true normal predictions, 3 false positives, 0 false negatives, and 20 true suspicious predictions. The recall is 100.00%, which means that all suspicious sessions in the simulation were detected. The trade-off is that three normal sessions were flagged as suspicious. In a cyber security environment, this trade-off is acceptable when analyst review is available because missing a real insider attack may be more harmful than investigating a small number of false alarms.

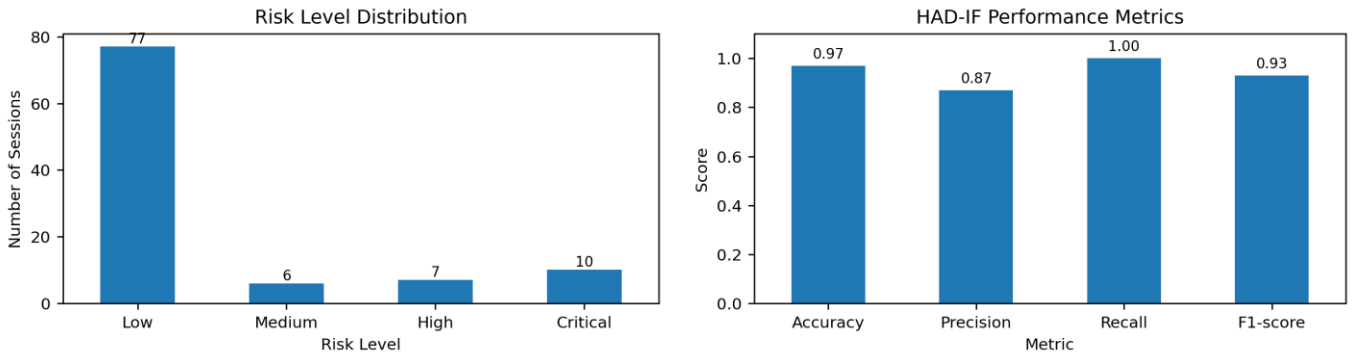



*Fig. 3. Risk-level distribution and performance metrics.*

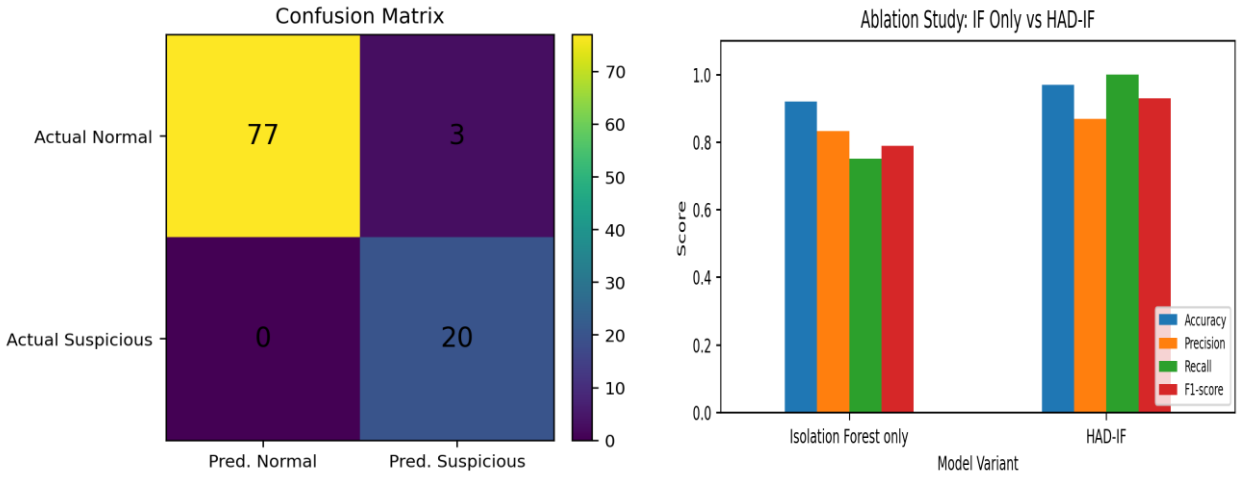



*Fig. 4. Confusion matrix and ablation comparison.*

The ablation chart shows the added value of the honeytoken signal. Isolation Forest only achieved 92.00% accuracy and 75.00% recall because it missed the stealth sessions whose behavior looked normal. HAD-IF improved the recall to 100.00% because honeytoken access marked those stealth sessions as suspicious. This result supports the main idea of combining behavioral AI with cyber deception.

## IX. Result Analysis

The results demonstrate that behavioral anomaly detection can correctly detect anomalous sessions where user behavior differs substantially from the baseline. Suspicious sessions displayed a higher volume of file accesses, higher volume of sensitive file access, more login failures and an order of magnitude greater volume of downloads-consistent with typical data-exfiltration and reconnaissance activities.

Honeytoken access greatly enhances the interpretability of the alert. A behavioral anomaly may be triggered during times of legitimately high work demand (e.g. Deadline, audit, emergency maintenance) or for innocent reasons such as development/testing. A user accessing a fake password file, or a fake confidential document carries greater security significance. It is for this reason that HAD-IF prioritizes honeytoken access as a Critical risk.

The ablation study presented here compares Isolation Forest alone to the full HAD-IF model. While Isolation Forest alone is informative, the honeytoken signal is critical to bolstering confidence and ensuring that traps are not ignored. This combination of AI and cyber deception is of the most practical use to the security field today.

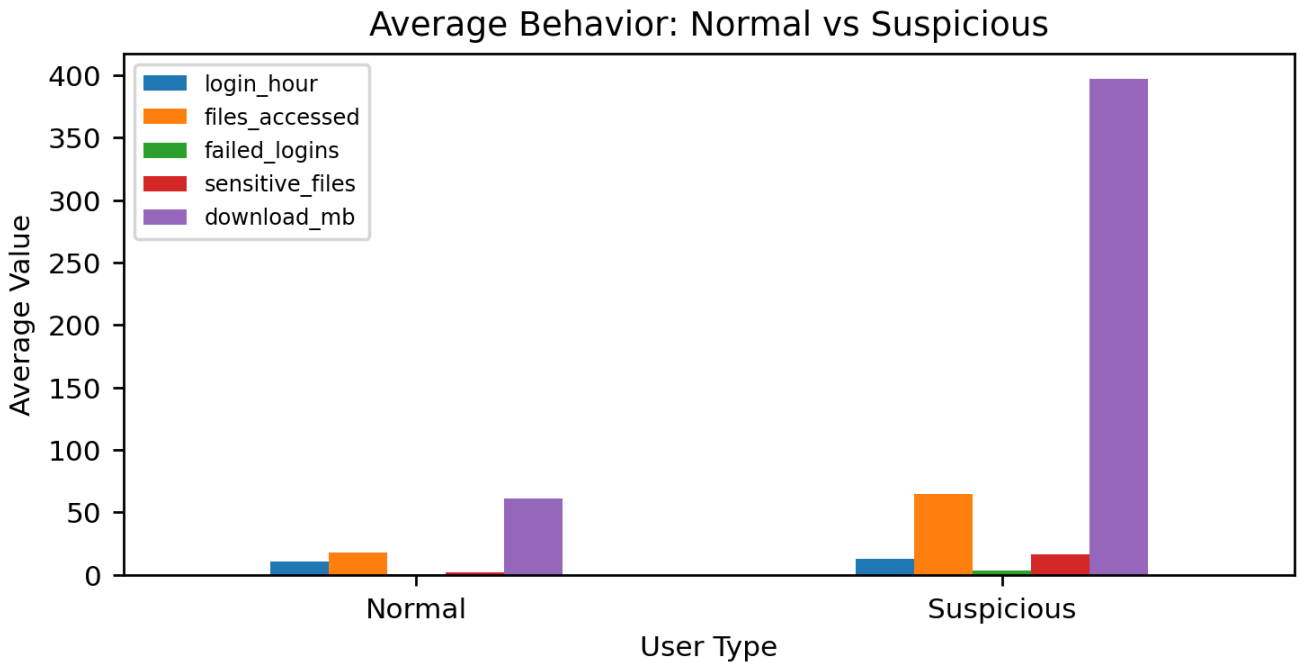



*Fig. 5. Average feature values for normal and suspicious user sessions.*

**Table 7 Average behavior comparison.**

| Type | Login Hour | Files | Failed | Sensitive | Download MB |
|---|---|---|---|---|---|
| Normal | 10.86 | 17.90 | 0.44 | 1.94 | 61.33 |
| Suspicious | 12.50 | 65.00 | 3.35 | 16.45 | 397.30 |

## X. Security, Privacy, and Ethical Considerations

Monitoring user behavior must be performed carefully. Organizations should define policies that explain what logs are collected, why they are collected, and how long they are retained. The goal of HAD-IF is defensive detection, not unauthorized surveillance. The system should respect privacy, legal requirements, and internal governance. Honeytokens should be designed so that they do not expose real secrets. A fake password file must never contain a real password. A fake customer database must not contain real customer information. The safest honeytokens are synthetic assets that look realistic but cannot damage real systems if accessed. Human review remains necessary. AI models may produce false positives, and unusual behavior may have legitimate causes. A critical alert should trigger investigation, evidence collection, and verification rather than automatic punishment. This aligns with responsible AI principles that require accountability, transparency, and oversight.

## XI. Limitations

First of all, the dataset is synthetically generated. While using a synthetic dataset is a good illustration of the idea behind the concept, it can't really mimic the complexity of an enterprise system. Real enterprise systems have varying user roles, departments, work times, projects, and access patterns.

Secondly, the feature space is too small. The model considers only login time, number of files, number of failed logins, number of accesses to sensitive files and number of downloads. In an enterprise system, features like the device name, IP address, country, session time, user role, user department, USB usage, e-mail, privilege level, and baseline (for users and the organization as a whole) are important.

Thirdly, the rule that defines risk is too simple. While this rule is easy to explain, it may not be the best rule for your environment. For example, your environment might benefit from a weighted risk score instead, a changing threshold or a user-role-specific baseline.

## XII. Future Work

For future research, the model should be tested on datasets like the CERT Insider Threat Test Dataset. This would enable a comparison with existing insider threat detection models. Cross-validation, class imbalance, and robustness tests should be included.

An addition that could be made to the model is explainable AI. For every alert, the system should produce reasons that lead to the alert. Examples are: login at 02:00, 70 files accessed, 7 failed logins, 25 sensitive files accessed, and honeytoken opened. These explanations would facilitate analysts in evaluating alerts.

Another aspect to investigate would be adaptive honeytoken deployment. Rather than providing every user with the same set of fake files, the system could generate honeytoken that would suit a user's specific department or job function. For example, a financial employee might find a dummy payroll file on his system, while a software developer might find a dummy api key file on his system. Synthetic and harmless dummy files would always have to be generated, however.
Lastly, it is desired to implement the model in a SIEM system. Alerts should be delivered on a dashboard, given severity ratings, linked to an incident ticket, and be examined by an analyst. The aforementioned integrations would enhance the applicability of the prototype to a real environment.

## XIII. Conclusion

The above work proposed HAD-IF, the Honeytoken-Augmented Deception Isolation Forest for insider-threat detection. This model takes an approach combining the strength of AI and cyber deception to better identify suspicious user actions. While Isolation Forest can identify anomalous sessions, the evidence of honeytoken usage serves as an additional indicator that, when received, is highly suggestive of suspicious user behavior. The final product assesses a session's activity as low, medium, high or

critically suspicious. Accuracy, precision, recall, and F1-score achieved from testing on synthesized data respectively were 97.00%, 86.96%, 100.00%, and 93.02%. The full detection recall means the simulation's suspicious sessions were all identified as suspicious. False positives shows that some human analysts would still be required. It must be said that this proposal should be considered an academic proof-of-concept and paper model; it should not be characterized as a "first ever" invention as both honeytokens and anomaly detection have appeared in the security literature before. The proposal's innovation comes in presenting a tangible framework combining anomaly detection, honeytoken use, and risk based alarming.

**References**

[1] C. Pascoe, S. Quinn, and K. Scarfone, “The NIST Cybersecurity Framework (CSF) 2.0,” National Institute of Standards and Technology, NIST CSWP 29, 2024, doi: 10.6028/NIST.CSWP.29.

[2] National Institute of Standards and Technology, “Insider Threat,” NIST Computer Security Resource Center Glossary, 2024.

[3] Cybersecurity and Infrastructure Security Agency, “Insider Threats 101 Fact Sheet,” CISA, 2024.

[4] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey,” ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, 2009, doi: 10.1145/1541880.1541882.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. 8th IEEE International Conference on Data Mining, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[6] L. Spitzner, “Honeypots: Catching the Insider Threat,” in Proc. 19th Annual Computer Security Applications Conference, 2003, pp. 170–179, doi: 10.1109/CSAC.2003.1254322.

[7] MITRE, “MITRE Engage: An Adversary Engagement Framework,” The MITRE Corporation, 2024.

[8] E. Tabassi, “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” National Institute of Standards and Technology, NIST AI 100-1, 2023, doi: 10.6028/NIST.AI.100-1.

[9] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[10] National Institute of Standards and Technology, “Insider Threat,” NIST Computer Security Resource Center Glossary, 2024.

[11] Cybersecurity and Infrastructure Security Agency, “Insider Threat Mitigation Guide,” CISA, 2020.

[12] Software Engineering Institute, “Common Sense Guide to Mitigating Insider Threats, Seventh Edition,” Carnegie Mellon University, 2022.

[13] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey,” ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, 2009, doi: 10.1145/1541880.1541882.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. 8th IEEE International Conference on Data Mining, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[15] MITRE, “MITRE Engage: An Adversary Engagement Framework,” The MITRE Corporation, 2024.

[16] L. Spitzner, “Honeypots: Catching the Insider Threat,” in Proc. 19th Annual Computer Security Applications Conference, 2003, pp. 170–179, doi: 10.1109/CSAC.2003.1254322.

[17] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, “Baiting Inside Attackers Using Decoy Documents,” in Security and Privacy in Communication Networks, SecureComm 2009, pp. 51–70, doi: 10.1007/978-3-642-05284-2_4.

[18] Cybersecurity and Infrastructure Security Agency, “Roadmap for Artificial Intelligence,” CISA, 2023.

[19] J. Kim, M. Park, H. Kim, S. Cho, and P. Kang, “Insider Threat Detection Based on User Behavior Modeling and Anomaly Detection Algorithms,” Applied Sciences, vol. 9, no. 19, Art. no. 4018, 2019, doi: 10.3390/app9194018.

[20] M. Akiyama, T. Hariu, T. Yagi, and Y. Kadobayashi, “HoneyCirculator: Distributing Credential Honeytoken for Introspection of Web-Based Attack Cycle,” International Journal of Information Security, vol. 17, pp. 621–634, 2018, doi: 10.1007/s10207-017-0361-5.

[21] N. Prabhaker, G. S. Bopche, and M. Arock, “Generation and Deployment of Honeytokens in Relational Databases for Cyber Deception,” Computers & Security, vol. 146, Art. no. 104032, 2024, doi: 10.1016/j.cose.2024.104032.