applied sciences

MDPI

Article

Mitigating Insider Threats Using Bio-Inspired Models

Andreas Nicolaou [1], Stavros Shiaeles [2,*] and Nick Savage [2]

[1] Computer and Network Security Research, Faculty of Pure and Applied Sciences, Open University of Cyprus, 2220 Nicosia, Cyprus; andreas.nicolaou@st.ouc.ac.cy

[2] School of Computing, Faculty of Technology, University of Portsmouth, Portsmouth PO1 2UP, UK; nick.savage@port.ac.uk

* Correspondence: stavros.shiaeles@port.ac.uk

Received: 23 June 2020; Accepted: 17 July 2020; Published: 22 July 2020

Featured Application: Authors are encouraged to provide a concise description of the specific application or a potential application of the work. This section is not mandatory.

Abstract: Insider threats have become a considerable information security issue that governments and organizations must face. The implementation of security policies and procedures may not be enough to protect organizational assets. Even with the evolution of information and network security technology, the threat from insiders is increasing. Many researchers are approaching this issue with various methods in order to develop a model that will help organizations to reduce their exposure to the threat and prevent damage to their assets. In this paper, we approach the insider threat problem and attempt to mitigate it by developing a machine learning model based on Bio-inspired computing. The model was developed by using an existing unsupervised learning algorithm for anomaly detection and we fitted the model to a synthetic dataset to detect outliers. We explore swarm intelligence algorithms and their performance on feature selection optimization for improving the performance of the machine learning model. The results show that swarm intelligence algorithms perform well on feature selection optimization and the generated, near-optimal, subset of features has a similar performance to the original one.

Keywords: bio-inspired computing; insider threats; swarm intelligence (SI); feature selection (FS); optimization; machine learning (ML); outlier detection; anomaly detection

1. Introduction

The recent Data Breach Investigations Report (DBIR) by Verizon reports that 34% of the reported data breaches were a result of internal actors' involvement and 2% of the data breaches were the result of a partner's involvement [1]. The report was based on an analysis of 41,686 security incidents, of which 2013 are confirmed data breaches. The previous yearly data breach reports, DBIR 2018 and DBIR 2017, show a data breach percentage involving internal actors of 28% and 25%, respectively [2,3]. The insider threat has been on the rise and the latest DBIR reports by Verizon confirm the rapid increase of the problem. The term "Data Breach" indicates a confirmed data disclosure after the event of a security incident [1]. An internal actor, or insider in an organization, is a current or former employee, partner, contractor, consultant, temporary personnel, personnel from partners, subsidiaries, contractors and anyone else that has been granted access privilege in the organization's network or data [4,5]. The Computer Emergency Readiness Team (CERT) National Insider Threat Center defines malicious insider as "a current or former employee, contractor, or business partner" who has authorized access to the organizational system and network resources and has intentionally exceeded or used that access in a manner that compromises the confidentiality, integrity and availability of the organization's data and information systems. An unintentional insider threat is an internal actor who has authorized

access to organizational system and network resources and "causes harm or substantially increases the probability of future serious harm of the confidentiality, integrity and availability of the organization's data and information systems" [6]. The insider threat is summed up as a security threat which describes the intentional or unintentional privileged misuse by an internal actor that causes damage to an organization's asset.

A survey conducted by the CERT National Threat Center and CSO Magazine revealed that 30% of the survey responders considered the damage caused by insider attacks more severe than the damage caused by outsider attacks [6]. Insider attacks include information system sabotage, theft of intellectual property, disclosure of confidential information, theft of trade secrets and espionage that leads organizations to financial losses and also negatively impacts their reputation and brand [6].

There is plenty of literature available on the mitigation of the insider threat problem that focuses on methods for detecting the insider threat. In this paper, we focus on the detection of insider threat using bio-inspired computing and utilizing machine learning.

Bio-inspired computing is an emerging approach, inspired by biological evolution, to develop new models that provides a solution for complex optimization problems in a timely manner. The explosion of data in the digital era has created challenges which are difficult to approach with traditional and conventional optimization algorithms and led the scientific community to develop bio-inspired algorithms that can be applied as a solution. Swarm Intelligence is a family of bio-inspired algorithms. These algorithms have been proposed by researchers to solve optimization problems by obtaining near-optimal solutions [7].

The purpose of this paper is to approach the insider threat problem with a new model that utilizes algorithms inspired by nature and contributes to the insider threat domain research by exploring metaheuristic algorithms to solve feature selection optimization problems and improve the performance of machine-learning-based insider threat detection models. The performance improvement of these models will help organizations and governments to detect malicious insiders in time, and prevent severe damage.

The rest of the paper is organized as follows. In Section 2, we review research related to the mitigation of insider threats. In Section 3, we present our methodology for the proposed approach. In Section 4, we present our findings from the evaluation of the algorithms and the improvement in the machine learning model after feature selection optimization. In Section 5, we discuss the results and findings. Finally, in Section 6, we conclude our findings and discuss future work.

2. Literature Review

The CERT division, part of Carnegie Mellon University's Software Engineering Institute, provides insider threat mitigation recommendations with the release of the "Common Sense Guide to Mitigating Insider Threats", based on research and analysis of previous insider threat cases. The guide includes and describes the practices that organizations should implement in order to reduce their exposure to the insider threat problem. Although this is the sixth edition of the guide, the insider threat problem continues to rise, which is another indication that further research must be made on the detection aspect of the problem [6]. Schultz [4] presents a framework based on insider behavior, to define insider-attack-related indicators and predict an attack. By using multiple and various indicators, there is a better chance of detecting or predicting the insider threat than using one [4]. While some indicators, such as "Preparatory behavior" for example, will indeed detect an insider attacker on the reconnaissance phase trying to gather information about the target, some others, such as "Meaningful Errors", depend on the attacker's skills and it will be hard to detect a skilful attacker.

Salem et al. [8] conducted research regarding the approaches and techniques for insider threat detection and acknowledge the challenge of building an effective and accurate system for detecting insider attacks. Brown et al. [9] propose a system to monitor electronic communication in an organization, to identify and predict an insider threat early. The system is based on personality factors and word correlations. It detects common words in the communication data and calculates

a score based on the predefined words' frequency of use. These scores are then combined into a composite personality factor score for neuroticism, agreeableness and conscientiousness, which are the three factors that are associated with high insider threat risk [9]. The authors state that their method mitigates possible legal or privacy concerns, but this was before the enforcement of GDPR. Monitoring electronic communication to profile a user is regulated by GDPR and raises privacy and legal issues. Axelrad et al. [10] propose a Bayesian network model, developed based on a list of variables associated with insider threats, to predict the potential malicious insider. The Bayesian network models generate a score for a person based on the person's characteristics. The list of variables was prepared after research through various papers addressing the insider threat problem. Correlations between variables were considered in the design of the model. Categories of variables include "personal life stressor and job stressors", personality and capability, attitude, workplace behavior and degree of interest [10]. As the authors acknowledge, there are some concerns regarding the collection of data for specific variables, such as job satisfaction, in which data may not be accurate. Nurse et al. [5] propose a framework to understand better and fully characterize the insider threat problem, developed after analysis of several real-world threat cases and the relevant literature. The authors' proposed unifying framework consists of several classes of components, which are presented in four main areas and broken down into more sections, beginning with the analysis of behavioral and psychological aspects related of the actor to understand one's tendency to attack. As the authors acknowledge, it is quite difficult to collect accurate psychological and historical behavioral information regarding insiders, to understand one's mind-set and, in many cases, this applies even after an attack. Behavioral analysis is continued in the next section as well by observing the physical and cyber behavior of the subject. Observing the physical and cyber behavior will be challenging to implement, since regulations vary among countries, for example in the European Union (EU), the General Data Protection Regulation (GDPR) regulates behavioral observation. Despite regulations, there are many challenges of monitoring the behavior of all insiders, for example contractors and partners. In the third section, the actor's type, enterprise role and state of relationship with the enterprise is defined, for example, whether the actor is an employee, contractor or partner, a current or former one and in what role he acts, as scientist, engineer, etc. The last two sections analyze the attack and the assets under the attack with their vulnerabilities. The proposed framework is indeed simple enough to follow, as the authors mention, and will help enterprises to analyze past attacks and identify weak points in their network, based on the insider attacker's steps [5]. Greitzer et al. [11], propose mitigation strategies and countermeasures for the unintentional insider threat, after their research of regarding cases and papers. The authors review possible causes and contributing factors and propose measures with an emphasis on employees' continuous training, to recognize threats such as phishing and enhance awareness of the insider threat problem. Mitigation strategies also include the enforcement of security policies and the implementation of security best practices, such as two-factor authentication. Although the proposed measures will enhance the awareness of the problem, they highly depend on the human factor and do not consider a change in an employee's behavior who might become an actual threat [11]. In order to build mechanisms for the detection and prevention of insider threats, real data need to be gathered and this "raises a variety of legal, ethical and business issues" [12].

Eldardiry et al. [13] propose a global model approach, based on feature extraction from user activities, based on a large amount of work practice data. This data is comprised of various domain areas, such as log files of logon and logoff events, HTTP browsing history, external device usage and file access. The authors evaluate their multi-domain system, utilizing ADAMS synthetic dataset to calculate the accuracy of anomalies and outlier detection and acknowledge that file access and external device usage domains can be used for easier threat prediction, compared to logon and HTTP history domains [13]. Rashid et al. [14] utilize Hidden Markov Models (HMM) with CERT's synthetic dataset to "learn" user normal behavior and then use HMM to detect significant changes in the "already learned" behavior. The authors report that their approach can be used to learn normal user's behavior, and then detect any significant deviations from it and detect potential malicious insiders, with high

accuracy. As the authors acknowledge, their model will not detect malicious insiders with no previously logged normal behavior, such as internal actors who attack an organization's systems, shortly after they log in. Lo et al. [15] apply Hidden Markov Method on CERT's synthetic dataset and analyze a number of distance measurement techniques, Damerau–Levenshtein Distance, Cosine Distance, and Jaccard Distance and their performance for detecting changes in user behavior. The authors report that, although HMM outscores each individual distance measurement technique, it needs more than a day to process all data. Le and Zincir-Heywood [16] propose a user-centered machine learning model that detects malicious insiders with high accuracy. The authors present a machine learning model focused on supervised learning by employing popular algorithms, such as Logistic Regressions (LR), Random Forest (RF) and Artificial Neural Network (ANN). Even though their proposed system detects malicious insiders with limited training, the authors propose the use of more sophisticated data pre-processing techniques and feature analysis to improve system performance. Liu et al. [17] acknowledge that, despite previous research on mitigating insider threats, organizations continue to report severe damage caused by malicious insiders. In their survey, they review several proposed systems addressing insider threats based on data analytics and identify relevant challenges. Log data analysis requires the collection of huge amounts of data, from a wide array of systems and a dedicated system to store the data for further processing. Since log data takes place on a variety of systems, there is no standard format for collecting log data and data pre-processing must be performed in order to clean the data and extract relevant features. As the authors acknowledge, this process requires extensive scripting and coding skills and a deep understanding of the various involved systems. Another challenging problem is extracting the essential and relevant features and managing them effectively by selecting the optimal subset to capture the attacker's footprint on time. Detecting the attacker's tiny footprint is like "find a needle in a haystack", and the challenge comes in deciding which method to utilize. The authors report that incorporating prior domain knowledge to a certain degree during the feature extraction process, may offer better results than entirely relying on prior domain knowledge.

In the literature reviewed for this paper, we came up with a limited study that uses bio-inspired computing to mitigate insider threats. Much research focuses on machine-learning-based insider threat detection to identify unusual behavior of users in regard to their normal behavior. Machine learning models rely on domain knowledge in the feature extraction and selection process, resulting in more time being consumed during data pre-processing and limited effectiveness in detecting the threats in cases where domain knowledge is not stated. Several researchers utilized bio-inspired models to address optimal solutions in complex problems. In this paper, we utilize bio-inspired computing to enhance machine learning models by automating the feature selection process and utilize unsupervised algorithms for outlier detection.

3. Methodology

Our proposed approach uses Swarm Intelligence Algorithms to automate feature selection optimization and eliminate unnecessary features before fitting the log data to a machine learning algorithm. Feature Selection or variable selection is the process of selecting the most relevant features for a specific problem and omitting unneeded or irrelevant and redundant features, to improve a model's prediction performance, reduce resource requirements, processing and utilization times [18]. We start by consolidating the log data from a synthetic dataset into a single data frame by parsing specific data such as the date into a more meaningful and useful data chunks, so that our algorithms can be more efficient. An automatic feature selection optimization, which is described in Section 3.3, is then applied to the generated data frame to produce the optimum subset of features. In Section 3.1, we present an overview of the proposed system, along with the data flow between the system components.

3.1. System Overview

In Figure 1, we illustrate an overview of our proposed system for malicious behavior detection. In the first step we collect data from various sources as well as the user information. Data Pre-processing takes place in the next step, along with all relevant log parsing. In the third step, feature selection optimization using bio-inspired models, is performed to generate the optimal features subset. In the fourth step, we fit the anomaly detection algorithm into the generated subset to detect outliers. In the final step, we analyze the results and measure the algorithm's performance.

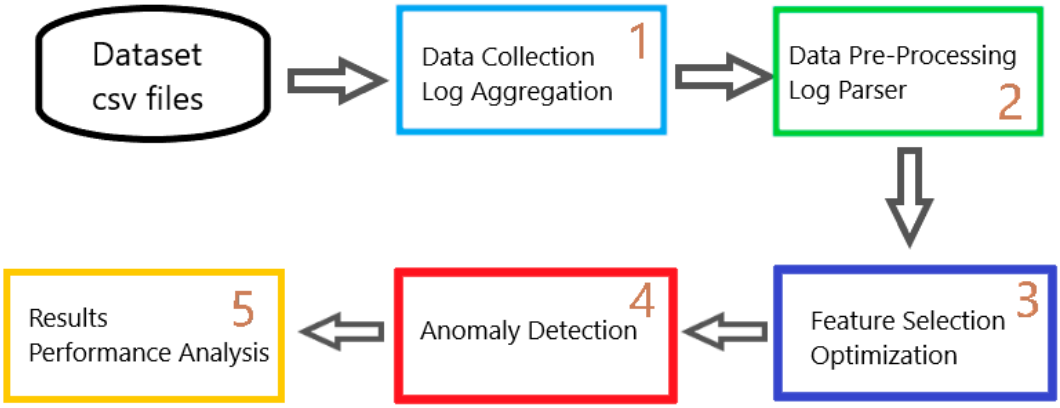

Figure 1. Diagram of proposed method and data flow inside the model.

3.2. Data Collection and Pre-Processing

Data are an essential element in threat detection models and play a crucial role in the detection of security incidents [19]. In order to avoid legal and privacy issues we are using a publicly available synthetic dataset that has no privacy constraints. The synthetic datasets contain logs generated specifically for insider threat research, and each dataset contains a small number of insider threat incidents regarding the dataset's accompanied scenario [12]. We need to parse records from the log data and extract data element values in a way that our model can use and make accurate predictions from the given data.

3.2.1. Dataset and Data Collection

The synthetic dataset that we evaluated in our system is a release from the Insider Threat Test Dataset collection, generated from Carnegie Mellon University Division. This dataset is "free of privacy and restriction limitations" [12] to allow researchers to experiment with it and evaluate algorithms. There are various releases of datasets to choose from, with most of them having one instance of each scenario, depending on the creation time. We chose the release r4.2, the "dense needle" dataset, since it includes many instances of each scenario, with multiple users involved in each scenario. The r4.2 dataset is split up in seven (7) different parts, as shown in Table 1.

Table 1. Insider threat dataset r4.2 files' description.

File	Description
Device.csv	Log of user's activity regarding connecting and disconnecting a thumb drive
Email.csv	Log of user's e-mail communication
File.csv	Log of user's activity regarding copying files to removable media devices
http.csv	Log of user's internet browsing history
Logon.csv	Log of user's workstation logon and logoff activity
psychometric.csv [(1)]	Users' psychometric scores based on big five personality traits
Ldap [(1)]	A set of eighteen (18) files regarding the organizations' users and their roles

[(1)] data not collected in our experiments.

We are not going to employee the data of psychometric.csv in our system, since it contains personality data and we want to avoid any privacy or legal issues regarding this in a real-life scenario. Psychometric.csv provides personality scores for each user, based on the big five personality traits. Since the used dataset is a synthetic one and there are no privacy or legal constraints in using this dataset's personality scores, we could add these features as well to experiment with. However, the scope of this work is intended to be used by organizations which might not have this kind of data. In addition to this, psychometric data are usually recorded by Human Resources (HR) and it will be difficult or even impossible to include scores for all internal actors, such as consultants, contractors, partners or even personnel from subsidiaries. Furthermore, we followed the Rashid et al. [14] approach and chose features that can be used to model user behavior across several different domains.

The Http.csv file contains data that can be used to trace employees' visits to employment websites and report indications for unsatisfied or even disgruntled employees that are planning to leave the company. While these data can be useful to predict an employee intention for leaving the company, privacy concerns arise. Furthermore, an additional overhead is added to the model, which eventually may not be effective in a real-world scenario where an employee can use their smartphone to access this kind of website.

3.2.2. Data Pre-Processing

Date values were collected as they were, but we had to split and encode the date feature into two features, day and time. Machine learning algorithms understand only integer numbers, so we had to convert the date and time into numbers and the other features as well, such as the activity. The activity feature's initial values correspond to user actions, such as logon to a system, logoff, connect thumb drive, disconnect, send an e-mail, process a file or access the internet. Since we have seven (7) different actions for the activity feature, we can replace each action with an integer, logon is 1, logoff is 2, etc. Our system's initial selected features are presented in Table 2, along with their value space.

Table 2. Encoded features at pre-processing phase.

Feature	Possible Values
Day	0, 1, 2, 3, 4, 5, 6
Time	1, 2, 3, 4, . . . , 24
User	String Type
PC	String Type
Activity [(1)]	1, 2, 3, 4, 5, 6, 7

[(1)] The values for the activity feature correspond to the user's activities, such as logon, logoff, connect, disconnect, HTTP, file and e-mail.

An example dataset comprised of the aforementioned features is shown in Table 3.

Table 3. Encoded features at pre-processing phase.

Day	Time	User	PC	Activity [(1)]
5	2	4512	4512	3

[(1)] We encoded categorical features using One-Hot Encoding scheme at a later stage since Machine learning algorithms are more effective in prediction when working with datasets encoded this scheme.

3.3. Feature Selection Optmization using Bio-Inspired Algorithms

For the purpose of this paper, we decided to use EvoloPy-FS framework [20–25] and measure the performance of several popular Swarm Intelligence algorithms on the feature selection optimization problem. EvoloPy-FS, an easy to use Python framework, developed by its authors to help researchers in solving optimization problems using Swarm intelligence algorithms. The main component of the framework is the Optimizer, where we set up our experiment along with the initial configurations.

In the optimizer, we define the dataset to use, the optimizers, the number of runs and number of iterations. For each implementation of the included optimizers, there is a separate Python script, since the framework is Open Source and all included components are transparent.

The Optimizers are used to generate a near-optimal subset of features from the original dataset, free from unnecessary features, to improve the model's anomaly prediction performance.

In our approach, for feature selection optimization we selected Binary Particle Swarm Optimization (BPSO), Binary Gray Wolf Optimizer (BGWO), Binary Bat Algorithm (BBAT), Binary Multi-Verse Optimizer (BMVO), Binary Moth-Flame Optimizer (BMFO), Binary Whale Optimization Algorithm (BWOA) and Binary Firefly Algorithm (BFFA) as optimizers, as they are available in EvoloPy-FS framework. We employed these bio-inspired models in the feature selection optimization process, to generate the optimal subset and fit the Machine Learning algorithm on it, to get better results compared with the original dataset.

3.4. Utilizing Machine Learning for Outlier Detection

Machine Learning (ML) is applied in a wide area of applications to discover patterns from given data and make predictions, such as anomaly detection. Several approaches, reported in the literature, utilize Machine Learning as an effective method for detecting anomalies. For the purpose of this paper, we developed a Machine Learning system, focused on user-centred analysis to distinguish malicious activities from the legitimate ones. The system utilizes the Local Outlier Factor (LOF), to detect the outlier or rare instances from the given data. LOF is fitted to the subset data frame, generated after feature selection optimization, to detect the outliers. For every detected outlier, the system marks the corresponding insider as malicious. The system marks an outlier based on the entire subset dataframe, based on the selected features and not based on CERT's accompanied scenarios.

Local Outlier Factor (LOF), is an unsupervised anomaly detection algorithm, which uses a score to determine if a certain point is an anomaly. Each datapoint is assigned this score, which is the result of the computation of local density deviation of the given datapoint with respect to its neighbor data oints. If a given datapoint has a substantially lower density than its neighbors, then it is a considered as an outlier [26]. The LOF algorithm is considered as an efficient method to detect outliers in high dimensional datasets. In our system, we employed sklearn.neighbors.LocalOutlierFactor from the scikit-learn module, which utilizes K-nearest neighbours to compute the local density of a datapoint. LOF score value is determined from the ratio of the average local density of the observation's neighbors and its own local density [27]. In LOF, we need to define the values of n_neighbors, contamination and n_jobs [27]:

- n_neighbors: the number of neighbors to take into consideration to detect the outliers. If the value is larger than the number of provided samples, then all samples will be used;
- Contamination: the proportion of outliers in the dataset. Contamination is used to define the threshold on the scores of the samples we fit LOF to;
- n_jobs: number of parallel jobs to run or neighbors search. "−1" value uses all available processors.

We fit LOF to the "candidate" optimal subset data frame, generated after performing feature selection optimization using bio-inspired models, to detect outliers.

3.5. Performance Metrics

To measure the performance of the subset dataset, generated after feature selection optimization, we followed Ferreira et al. [28] performance metrics' method, who used the insider detection rate and insider detection precision. As mentioned in Section 3.4, the subset dataset is fitted to LOF to detect anomalies and we measure the performance of the results for each subset dataset tested, to find the

optimal subset. The subset dataset that produces the best precision will be selected as the optimal one. Insider detection rate (DR) and precision can be determined by using Equations (1) and (2).

$$DR = \frac{TP}{TP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

In Equations (1) and (2) True Positive (TP), represents the number of true malicious users detected, False Positive (FP) represents the number of normal users that were detected as malicious, and False Negative (FN) represents the malicious users that were not detected as malicious, but falsely considered as normal users.

4. Results

In this section, we present the results from our experiments, along with the various scenarios we executed to test the model and measure the performance of the utilized swarm intelligence algorithms for feature selection optimization.

4.1. Experimental Setup

All data processing tasks for this paper are performed using a PC with Intel Core ™ i5 4200M @ 2.5GHz CPU and 16.0 GB Dual-Channel DDR @798MHz RAM. All algorithms are tested using Anaconda's Python distribution version 2019.07. The global settings are the same for all Swarm Intelligence algorithms in order to have fair comparisons. Population size is set to 50 search agents, and the number of iterations is set to 20. For the purpose of this paper, we used and utilized Python programming language along with several open-source libraries:

- Anaconda;
- Python;
- Jupyter;
- Pandas;
- Numpy;
- EvoloPy-FS;
- Scikit-learn.

4.2. Testing the Model

The objective of the test is to improve the performance of the Machine learning model by obtaining the optimal subset, using Bio-inspired models, before fitting Local Outlier Factor (LOF) algorithm to it and determine the outliers.

We created samples of the first 50,000 rows from the synthetic dataset, to work with a smaller portion of it, for performance reasons. Sample creation code is presented in Figure 2.

```
#create samples for our experiment
sampleLogins = pd.read_csv('d:\\master\\r4.2\\logon.csv', low_memory=False, nrows=50000)
sampleLogins.to_csv('d:\\master\\r4.2\\sample-logon.csv', index=False)

sampleDevices = pd.read_csv('d:\\master\\r4.2\\device.csv', low_memory=False, nrows=50000)
sampleDevices.to_csv('d:\\master\\r4.2\\sample-device.csv', index=False)

sampleFiles = pd.read_csv('d:\\master\\r4.2\\file.csv', low_memory=False, nrows=50000)
sampleFiles.to_csv('d:\\master\\r4.2\\sample-file.csv', index=False)

sampleFiles = pd.read_csv('d:\\master\\r4.2\\http.csv', low_memory=False, nrows=50000)
sampleFiles.to_csv('d:\\master\\r4.2\\sample-http.csv', index=False)

sampleFiles = pd.read_csv('d:\\master\\r4.2\\email.csv', low_memory=False, nrows=50000)
sampleFiles.to_csv('d:\\master\\r4.2\\sample-email.csv', index=False)
```

Figure 2. Create samples from the synthetic dataset.

Data collection and data pre-processing code execution take place in order to prepare the data frame with all selected features and fit the feature selection optimization to it.

We fitted the feature selection optimization framework on the first thousand (1000) rows of the generated data, with the results shown in Table 4. The seven (7) columns represent the results for each optimizer from the selected, BPSO, BMVO, BGWO, BMFO, BWOA, BFFA and BBAT. Time taken is in seconds and is about the same for all seven optimizers. Train and testing accuracy are computed based on the sliced part for each value, as shown in Figure 3 and against the complete dataset.

Table 4. Feature selection optimization results.

	BPSO	BMVO	BGWO	BMFO	BWOA	BFFA	BBAT
Time Taken [1]	9.4016	11.3000	8.8428	8.5710	12.0246	10.5799	9.3706
Train Accuracy [1]	0.9985	0.9939	0.9939	0.9939	0.9939	0.9939	0.8030
Test Accuracy [1]	0.9941	1	1	1	1	1	0.8235
Number of Iterations x Features Selected [1]	7 × 4 Feat. 4 × 5 Feat. 2 × 6 Feat. 4 × 7 Feat.	2 × 4 Feat. 15 × 5 Feat. 2 × 6 Feat.	0 × 4 Feat. 19 × 5 Feat.	0 × 4 Feat. 19 × 5 Feat.	2 × 4 Feat. 15 × 5 Feat. 3 × 6 Feat.	3 × 4 Feat. 14 × 5 Feat. 0 × 6 Feat. 3 × 7 Feat.	5 × 4 Feat. 8 × 5 Feat.

[1] Results of a single independent run executed for each algorithm.

```
DatasetSplitRatio=0.34    #Training 66%, Testing 34%
```

Figure 3. Training % | Testing %.

In order to test the feature selection output results if they generate an optimal subset, we need to fit LOF to the subset generated after the feature selection. The performance can be measured by comparing time taken for insider threat detection along with the detection rate and precision. Precision is the ratio of True Positives to the total number of positive results, Precision = TPP / (TP + FP), with 1 being the best value and 0 the worst.

The first measurement is done by selecting all features (Figure 4), as generated after the pre-processing data phase, and see the results without feature selection optimization. Following this, we run more experiments based on the results of feature selection optimization. The results of the experiments are presented in Table 5.

```
#select all features
selected_features = ['day', 'time', 'Logon', 'Logoff', 'Connect', 'Disconnect', "email", "file", "http"]
```

Figure 4. All features selected.

Table 5. LOF results with feature optimization. N_neighbors = 20 and contamination = auto.

Experiment [1]	Features	Time_Taken	TP	FP	FN	DR	Precision
1	9.0	8.604782	70.0	918.0	0.0	1.000000	0.070850
2	5.0	6.609488	69.0	910.0	1.0	0.985714	0.070480
3	4.0	5.928126	62.0	774.0	8.0	0.885714	0.074163
4	7.0	7.686987	70.0	918.0	0.0	1.000000	0.070850
5	6.0	6.994491	69.0	919.0	1.0	0.985714	0.069838

[1] In all experiments, a portion of the original synthetic dataset was used to detect outliers.

The results of Table 5 show a high detection rate but very low precision, since we have a high number of FP. We can experiment with the parameters of LOF algorithm and change n_neighbors and contamination values to see whether we can get improved precision results. In order to get the results in Table 5, n_neighbors' value was set to 20 and contamination set to auto.

The performance of LOF is highly dependent on the values of contamination and n_neighbors [29]. We set the value of n_neighbors to 20, which is the default value of the utilized Machine Learning algorithm [27] and defines the number of neighbors that need to be taken into consideration to detect the outliers. As mentioned in Section 3.4, contamination represents the proportion of outliers in the dataset and its value defines the number of objects to be predicted as anomalies. Contamination can be set as a float number with a value between 0 and 0.5 [27]. The higher the value, the more objects will be predicted as anomalies. Since we had a high number of FP in the first run of the model (Table 5), we continued our experiments by reducing the value of contamination to reduce the number of FP.

We changed the contamination value to 0.1, to get similar results when contamination was set to auto (results are shown in Table 6). The optimal subset remains the same, but precision value is still very low.

Table 6. Local outlier factor (LOF) results with feature optimization. N_neighbors = 20 and contamination = 0.1.

Experiment [1]	Features	Time_Taken	TP	FP	FN	DR	Precision
1	9.0	11.854724	70.0	925.0	0.0	1.000000	0. 070352
2	5.0	6.610281	69.0	923.0	1.0	0.985714	0. 069556
3	4.0	5.990587	65.0	837.0	5.0	0. 928571	0. 072062
4	7.0	7.884163	70.0	925.0	0.0	1.000000	0. 070352
5	6.0	8.453545	70.0	927.0	0.0	1.000000	0. 070211

[1] In all experiments, a portion of the original synthetic dataset was used to detect outliers.

By changing the contamination value to 0.01 we got slightly different results (shown in Table 7) compared with the previous two cases and, in this case, the better precision value is produced by experiment #2, with a subset dataset of five (5) features. This subset dataset was produced by all seven (7) algorithms during most of the algorithms' iterations.

Table 7. LOF results with feature optimization. N_neighbours = 20 and contamination = 0.01.

Experiment [1]	Features	Time_Taken	TP	FP	FN	DR	Precision
1	9.0	9.417012	59.0	658.0	11.0	0.842857	0.082287
2	5.0	6.663487	56.0	573.0	14.0	0.800000	0.089030
3	4.0	5.407866	55.0	645.0	15.0	0.785714	0.078571
4	7.0	8.115871	58.0	654.0	12.0	0.828571	0.081461
5	6.0	7.711550	56.0	730.0	14.0	0.800000	0.071247

[1] In all experiments a portion of the original synthetic dataset was used to detect outliers.

We continued our experiments by tuning the contamination value in LOF and fitting the LOF algorithm on the optimized subset of four (4) features. In Table 8, the reported results indicate that, for the specific synthetic dataset, we can get better results with lower contamination values.

Table 8. LOF results for subset of four (4) features.

Experiment [(1)]	Contamination	Time_Taken	TP	FP	FN	DR	Precision
1	0.001	488	44.0	252.0	26.0	0.628571	0.148649
2	0.002	516	58.0	339.0	12.0	0.828571	0.146096
3	0.003	480	62.0	394.0	8.0	0.885714	0.135965
4	0.004	488	63.0	414.0	7.0	0.9	0.132075
5	0.005	481	64.0	421.0	6.0	0.914286	0.131959

[(1)] In experiments 1-5 we utilized full data from logins, devices and files dataset files.

The performance of an LOF algorithm depends on its parameters' values, contamination and neighborhood size [29]. When experimenting with synthetic data with the known anomaly portion, contamination value and neighborhood size can be tuned based on this known anomaly portion data and report better results. In a real-world scenario, these parameters can be tuned based on historic evidence of malicious insiders or use the Xu et al. [29] methodology for automatic tuning Local Outlier Factor's hyperparameters.

5. Discussion

The findings of Section 4.2, show that after feature selection optimization, one of the resulted subsets is a near-optimal subset, since it performs better than the original one when measured with precision. This near-optimal subset was generated after feature selection optimization by using BPSO, BMVO, BWOA, BFFA, BBAT optimizers.

These results acknowledge that Swarm intelligence algorithms have high performance on feature selection optimization problems and can be used to enhance Machine Learning models. The use of bio-inspired models in our proposed system resulted in better precision value when utilized in the feature selection optimization process, before fitting the machine learning algorithm to the dataset.

We compared our approach with approaches that used Hidden Markov Models (HMM), Damerau–Levenshtein (DL) Distance, Cosine Distance, and Jaccard Distance techniques (Table 9 reports the performance of each approach based on TP/FP detection rate).

Table 9. Comparison with other approaches.

Author	Approach	TP	FP
This paper	Table 8—Exp.5	91.4%	45%
This paper	Table 8—Exp. 2	83%	36%
[14]	HMM	85%	20%
[15] [(1)]	DL	39%	
[15] [(1)]	Jaccard	36%	
[15] [(1)]	Cosine	47%	
[15] [(1)]	DL, Jaccard & Cosine aggregate score	80%	
[15] [(1)]	HMM	69%	

[(1)] The authors report a potential of a high number of FP but do not mention the number of FP in their results.

Rashid et al. [14] used HMM and reported an 85% identification rate with a false positive rate of 20%. Lo et al. [15] acknowledge that during the training phase of HMM, the computational time can be quite slow, as the number of features increases. Lo et al. [15] used HMM and distance measurement techniques and reported detection rates of 69% and 80% (aggregate score), respectively. The authors reported that HMM took more than 24 h to process all data, in opposition to distance measurements that took minutes to process. While the authors report that the combination of the three distance

measurements techniques has the potential of raising a high number of FP, they do not mention the number of FP of their results. While our approach reports a high number of FP, it also reports a better TP identification rate compared with other approaches, as reported in Table 9.

6. Conclusions

In this paper, we introduce the use of bio-inspired computing in machine learning models for mitigating insider threats and we improve the model by automating the feature selection optimization process. We evaluate several swarm intelligence algorithms and our results show that swarm intelligence algorithms should be employed to improve accuracy and speed in detecting malicious behavior in large data sets.

An optimal subset with reduced features has similar or better performance to the original one and can improve the performance of a machine learning model.

The employment of labeled data and the addition of extra features, such as an indication of visits to employment websites, social networking sites and cloud storage services, where an internal actor can share confidential information to others will improve the performance of our system in the detection of malicious insiders and reduce the FP rate. These additional indicators must be transparent to all internal actors, who must be clearly informed about all their log activity, according to relevant privacy regulation. The collection of an employee's private data in an organization might raise several issues and concerns and eventually reduce productivity.

The usage of unsupervised ML techniques to detect anomalies using unlabeled data, not only protects the privacy of legitimate internal actors, but also detects anomalies in the behavior of malicious insiders with no previous logged history, since no prior training is needed for unsupervised learning.

6.1. Limitations

The process of detecting malicious insiders by detecting significant changes, or anomalies in a user's normal behavior, might be inconsistent in several circumstances, such as a team of employees working overtime on a project with a strict deadline, or another team resolving a system failure during work after hours. Regarding after-hours, certain employees might work on a rotating shift schedule, thus detecting outliers based on normal and after-hours' activity may not work for them.

6.2. Future Research

For future work, we can fit our proposed system to other existing or future releases of CERT's datasets, experiment with other bio-inspired models, such as cuckoo search (CS) and compare their performance with the bio-inspired models we experiment within this paper.

In future works, similar systems should be developed and evaluate hybrid algorithms or explore the possibility of detecting anomalies with the use of bio-inspired computing.

Author Contributions: All authors contributed equally. Conceptualization, S.S., A.N.; methodology, A.N, S.S.; software, A.N.; validation, A.N., S.S., N.S.; investigation, A.N.; resources, A.N.; writing-original draft preparation, A.N.; writing-review and editing, S.S. and N.S.; supervision, S.S. All authors have read and agreed to the published version of the manuscript. The authors read and approved the final manuscript as well as the authors order.

Funding: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 833673. This work reflects authors' view and the agency is not responsible for any use that may be made of the information it contains.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Verizon Enterprise. Verizon Data Breach Investigations Report 12th Edition. Available online: https://www.verizonenterprise.com/verizon-insights-lab/dbir/ (accessed on 30 August 2019).
2. Widup, S.; Spitler, M.; Hylender, D.; Bassett, G. Verizon Data Breach Investigations Report 11th Edition. Available online: https://www.verizonenterprise.com/verizon-insights-lab/dbir/ (accessed on 9 February 2019).
3. Verizon Enterprise. Verizon Data Breach Investigations Report 10th Edition. Available online: https://www.verizonenterprise.com/verizon-insights-lab/dbir/ (accessed on 30 August 2019).
4. Schultz, E.E. A framework for understanding and predicting insider attacks. *Comput. Sec.* **2002**, *21*, 526–531. [CrossRef]
5. Nurse, J.R.C.; Buckley, O.; Legg, P.A.; Goldsmith, M.; Creese, S.; Wright, G.R.T.; Whitty, M. Understanding insider threat: A framework for characterising attacks. In Proceedings of the 2014 IEEE Security and Privacy Workshops, San Jose, CA, USA, 17–18 May 2014; pp. 214–228. [CrossRef]
6. Theis, M.; Trzeciak, R.; Costa, D.; Moore, A.; Miller, S.; Cassidy, T.; Claycomb, W. Common Sense Guide to Mitigating Insider Threats, Sixth Edition. Available online: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=540644 (accessed on 9 February 2019).
7. Krishnanand, K.R.; Nayak, S.K.; Panigrahi, B.K.; Rout, P.K. Comparative study of five bio-inspired evolutionary optimization techniques. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 1231–1236. [CrossRef]
8. Salem, M.B.; Hershkop, S.; Stolfo, S.J. A survey of insider attack detection research. In *Insider Attack and Cyber Security*; Springer: Boston, MA, USA, 2008; pp. 69–90. [CrossRef]
9. Brown, C.R.; Watkins, A.; Greitzer, F.L. Predicting insider threat risks through linguistic analysis of electronic communication. In Proceedings of the 2013 46th Hawaii International Conference on System Sciences, Wailea, Maui, HI, USA, 7–10 January 2013; pp. 1849–1858. [CrossRef]
10. Axelrad, E.T.; Sticha, P.J.; Brdiczka, O.; Shen, J. A Bayesian Network Model for Predicting Insider Threats. In Proceedings of the 2013 IEEE Security and Privacy Workshops, San Francisco, CA, USA, 23–24 May 2013; pp. 82–89. [CrossRef]
11. Greitzer, F.L.; Strozer, J.; Cohen, S.; Bergey, J.; Cowley, J.; Moore, A.; Mundie, D. Unintentional Insider Threat: Contributing Factors, Observables, and Mitigation Strategies. In Proceedings of the 2014 47th Hawaii International Conference on System Sciences, Waikoloa, HI, USA, 6–9 January 2014; pp. 2025–2034. [CrossRef]
12. Glasser, J.; Lindauer, B. Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. In Proceedings of the 2013 IEEE Security and Privacy Workshops, San Francisco, CA, USA, 23–24 May 2013; pp. 98–104. [CrossRef]
13. Eldardiry, H.; Bart, E.; Liu, J.; Hanley, J.; Price, B.; Brdiczka, O. Multi-Domain Information Fusion for Insider Threat Detection. In Proceedings of the 2013 IEEE Security and Privacy Workshops, San Francisco, CA, USA, 23–24 May 2013; pp. 45–51. [CrossRef]
14. Rashid, T.; Agrafiotis, I.; Nurse Jason, R.C. A New Take on Detecting Insider Threats: Exploring the Use of Hidden Markov Models. In Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats (MIST'16), Vienna, Austria, 24–28 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 47–56. [CrossRef]
15. Lo, O.; Buchanan, W.J.; Griffiths, P.; Macfarlane, R. Distance measurement methods for improved insider threat detection. *Sec. Commun. Netw.* **2018**, *2018*, 18. [CrossRef]
16. Le, D.C.; Zincir-Heywood, A. Nur. Machine learning based Insider Threat Modelling and Detection. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 8–12 April 2019; pp. 1–6.
17. Liu, L.; De Vel, O.; Han, Q.; Zhang, J.; Xiang, Y. Detecting and Preventing Cyber Insider Threats: A Survey. In *IEEE Communications Surveys & Tutorials*; IEEE: New York, NY, USA, 2018; Volume 20, pp. 1397–1417. [CrossRef]
18. Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.

19. Sun, N.; Zhang, J.; Rimba, P.; Gao, S.; Zhang, L.Y.; Xiang, Y. Data-Driven Cybersecurity Incident Prediction: A Survey. In *IEEE Communications Surveys & Tutorials*; IEEE: New York, NY, USA, 2019; Volume 21, pp. 1744–1772. [CrossRef]
20. Khurma, R.A.; Aljarah, I.; Sharieh, A.; Mirjalili, S. EvoloPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection. In *Evolutionary Machine Learning Techniques*; Springer: Singapore, 2020; pp. 131–173. [CrossRef]
21. Faris, H.; Aljarah, I.; Mirjalili, S.; Castillo, P.A.; Guervós, J.J.M. EvoloPy: An Open-source Nature-inspired Optimization Framework in Python. In Proceedings of the 8th International Joint Conference on Computational Intelligence IJCCI (ECTA), Porto, Portugal, 9–11 November 2016; pp. 171–177.
22. Faris, H.; Heidari, A.A.; Ala'M, A.-Z.; Mafarja, M.; Aljarah, I.; Eshtay, M.; Mirjalili, S. Time-varying hierarchical chains of salps with random weight networks for feature selection. *Expert Syst. Appl.* **2020**, *140*, 112898. [CrossRef]
23. Mafarja, M.; Aljarah, I.; Heidari, A.A.; Faris, H.; Fournier-Viger, P.; Li, X.; Mirjalili, S. Binary dragonfly optimization for feature selection using time-varying transfer functions. *Knowl. Based Syst.* **2018**, *161*, 185–204. [CrossRef]
24. Aljarah, I.; Mafarja, M.; Heidari, A.A.; Faris, H.; Zhang, Y.; Mirjalili, S. Asynchronous accelerating multi-leader salp chains for feature selection. *Appl. Soft Comput.* **2018**, *71*, 964–979. [CrossRef]
25. Faris, H.; Mafarja, M.M.; Heidari, A.A.; Aljarah, I.; Ala'M, A.-Z.; Mirjalili, S.; Fujita, H. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowl. Based Syst.* **2018**, *154*, 43–67. [CrossRef]
26. Breunig, M.M.; Kriegel, H.-P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 15–18 May 2000; pp. 93–104. [CrossRef]
27. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
28. Ferreira, P.; Le, D.C.; Zincir-Heywood, N. Exploring Feature Normalization and Temporal Information for Machine Learning Based Insider Threat Detection. In Proceedings of the 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 21–25 October 2019; pp. 1–7. [CrossRef]
29. Xu, Z.; Kakde, D.; Chaudhuri, A. Automatic Hyperparameter Tuning Method for Local Outlier Factor, with Applications to Anomaly Detection. In Proceedings of the 2019 IEEE International Conference on Big Data, Los Angeles, CA, USA, 9–12 December 2019; pp. 4201–4207. [CrossRef]