Open University of Cyprus

School of Pure and Applied Sciences

Postgraduate Thesis
In Computer and Network Security

Mitigating Insider Threats using Bio-Inspired Models

Andreas S. Nicolaou

Supervising Professor
Dr. Stavros Shiaeles

May 2020

Open University of Cyprus

School of Pure and Applied Sciences

Mitigating Insider Threats Using Bio-Inspired Models

Andreas S. Nicolaou

Supervising Professor
Dr. Stavros Shiaeles

A thesis submitted in partial fulfillment for the postgraduate degree in

MSc Computer and Network Security

To the School of Pure and Applied Sciences
Open University of Cyprus

May 2020

BLANK PAGE

Summary

Insider Threat has become a huge 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 insider threat problem is on the rise and many researchers are approaching the problem 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 M.Sc. dissertation 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 explored swarm intelligence algorithms and their performance on feature selection optimization for improving the performance of the machine learning model. The results showed that swarm intelligence algorithms perform well on feature selection optimization and the generated near-optimal subset of features that has similar performance with the original one.

Περίληψη

Οι εσωτερικές απειλές έχουν εξελιχθεί σε ένα πολύ σοβαρό ζήτημα ασφάλειας πληροφοριών που έχουν να αντιμετωπίσουν κυβερνήσεις και οργανισμοί. Η εφαρμογή πολιτικών και διαδικασιών ασφαλείας ενδέχεται να μην είναι αρκετή για προστασία τα αγαθά ενός οργανισμού. Ακόμη και με την εξέλιξη της τεχνολογίας στον τομέα της κυβερνοασφάλειας, το πρόβλημα της εσωτερικής απειλής αυξάνεται και η ερευνητική κοινότητα προσεγγίζει το πρόβλημα με διάφορες μεθόδους, προκειμένου να αναπτύξει ένα μοντέλο που θα βοηθήσεις τους οργανισμούς να μειώσουν την έκθεση τους στην απειλή και να αποτρέψουν ζημιές στα περιουσιακά τους στοιχεία.

Με αυτή τη μεταπτυχιακή διατριβή προσεγγίζουμε το πρόβλημα προσεγγίζουμε το πρόβλημα της εσωτερικής απειλής αναπτύσσοντας μοντέλο μηχανικής μάθησης, χρησιμοποιώντας Bio-Inspired computing. Το μοντέλο αναπτύχθηκε χρησιμοποιώντας αλγόριθμο μάθησης χωρίς επίβλεψη για ανίχνευση ανωμαλιών και προσαρμόσαμε το μοντέλο σε ένα γνωστό συνθετικό σύνολο δεδομένων για εντοπισμών των ανωμαλιών. Εξερευνήσαμε Swarm intelligence αλγορίθμους και την απόδοση τους σε βελτιστοποίηση επιλογής χαρακτηριστικών για τη βελτίωση της απόδοσης του μοντέλου μηχανικής μάθησης. Τα αποτελέσματα έδειξαν ότι οι Swam intelligence αλγόριθμοι έχουν πολύ καλή απόδοση στη βελτιστοποίηση επιλογής χαρακτηριστικών και το near-optimal υποσύνολο χαρακτηριστικών έχει παρόμοια απόδοση με το αρχικό.

Ευχαριστίες | Acknowledgements

I would like to thank my wife Maria who encouraged me to apply for this program study and supported me through the whole time and my two sons, Savvas and Ioannis for their understanding and their patience during my journey of completing this work.

I would also like to thank my supervisor Dr. Stavros Shiaeles for trusting me with this M.Sc. dissertation, but also for his support on this work.

Table of Contents

Chapter 1
Introduction

The recent Data Breach Investigations Report (DBIR) by Verizon, reports that 34% of the reported data breaches was a result of internal actors' involvement and 2% of the data breaches was a result of partners involvement (Verizon, 2019). The report was built with an analysis of 41,686 security incidents, of which 2,013 are confirmed data breaches. The previous yearly data breaches reports, DBIR 2018 and DBIR 2017, show a data breach percentage involving internal actors of 28% and 25%, respectively (Widup et al., 2018; Verizon, 2017). Insider Threat has been on the rise and the latest DBIR reports by Verizon confirm the rapid increase of the problem.

In order to justify DBIR report's results to the reader, we first need to clarify a few terms mentioned in the report. A security incident is an event which compromises the confidentiality, integrity and availability of an information asset. With the term "Data Breach" we mean that after the event of a security incident, there was a confirmed data disclosure (Verizon, 2019). Confidentiality, Integrity and Availability, also known as CIA triangle or triad, is an Information Assurance model, designed years ago, to guide policies for Information Security in an organization. Confidentiality ensures that only authorized users can access the data, information and services of a system and that the communication between the user and the system remains private and usually is encrypted. Integrity ensures that data and information of data can be created, modified and deleted only by

authorized users. Availability ensures that data, services and information of a system is always available to legitimate users (Pfleeger and Pfleeger, 2002).

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 (Schultz, 2002; Nurse et al., 2014). A threat is any intentionally or unintentionally act, that exploits a vulnerability and cause damage to organization's assets. An asset can be any element of an Information System, such as software, hardware, data, procedures, communication or people. A vulnerability is a weakness of an asset.

The CERT National Insider Threat Center defines malicious insider as "a current or former employee, contractor, or business partner" who has authorized access to 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 series harm of the confidentiality, integrity and availability of the organization's data and information systems" (Theis at al., 2019).

Insider Threat is summed up as a security threat which describes the intentional or unintentionally privilege 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% the survey responders considered the damage caused by insider attacks more severe than the damage caused by outsider attacks (Theis at al., 2019). Insider attacks incidents include information system sabotage, theft of intellectual property, disclosure of confidential information, theft of trade secrets, espionage that leads organizations to financial losses but also negatively impact their reputation and brand (Theis at al., 2019).

1.1 Mitigating Insider Threats

To mitigate something means to make it less harmful and less severe (Dictionary.cambridge.org, 2019). By mitigating Insider Threats, we mean to establish security measures that will contribute

in the detection of the threats in an accurate and timely manner and respond accordingly in order to reduce the damage of the insider attackers.

In order to mitigate and combat Insider Threats, we first need to identify the threats and know our enemy. As Sun Tzu writes in his book "…if you know the enemy and know yourself, you need not fear the result of a hundred battles…" (Tzu, 2007). By knowing our enemies, we can improve our security measures to be more effective against them.

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

1.2 Bio-Inspired Computing

Bio-Inspired computing is an emerging approach, inspired by biological evolution, to develop new models that provides solution for complex optimization problems in a timely manner. The explosion of data in the digital era has created challenges difficult to approach with traditional and conventional optimization algorithms and lead the scientific community to develop Bio-Inspired algorithms that can be applied as a solution. Swarm Intelligence is a family of Bio-Inspired Algorithms (Abraham 2008). These algorithms have been proposed by researchers to solve optimization problems by obtaining near optimal solutions (Krishnanand, Nayak, Panigrahi and Rout, 2009).

1.3 Machine Learning

Machine Learning (ML) is a subset field Artificial Intelligence (AI) where we feed data into a model, to discover patterns from the given data and make predictions. Machine Learning is applied in a wide area of applications, such as healthcare, finance, biology and cybersecurity. Several algorithms are used in Machine Learning and are divided into three main categories, Supervised learning, Unsupervised learning and Reinforcement learning. In Supervised learning we are using labeled data to train the machine based on input and output data. In Unsupervised learning we can feed the machine with unlabeled data, since the objective is to detect regularities in the input data. The objective is to detect the patterns that occur more often than others. In Reinforcement learning a reward policy is involved and the machine learns to take decisions through trial and error

(Alpaydin, 2014). For the purpose of this M.Sc. dissertation we utilize Unsupervised learning for outlier detection as described in chapter 4.

1.4 Problem Statement

The purpose of this M.Sc. dissertation is to approach the Insider Threat problem with a new model that utilizes algorithms inspired by nature and contribute to the Insider Threat domain research by exploring meta heuristic algorithms to solve feature selection optimization problems and improve the performance of machine learning based insider threat detection models. The improvement of the performance of insider threat detection models will help organizations to detect malicious insiders in time before causing severe damage.

1.5 Chapters' Overview

In this chapter we introduced the reader to the insider threat problem, by defining the problem and related terms and we also state the importance of mitigating the insider threat problem.

In chapter 2 we review research related with the mitigation of insider threats and summarize previous research focused on computation intelligence inspired by nature. In chapter 3 we present an overview of Swarm Intelligence and Bio-Inspired computing and present several popular models. In chapter 4 we present our methodology for the proposed approach. Chapter 5 presents our proposed Insider Threat predicting model along with all the performed data pre-processing, feature extraction and feature selection optimization steps. In chapter 6 we present our findings from the evaluation of the algorithms and the improvement of the machine learning model after feature selection optimization and we also compare the performance of our approach with other approaches. Finally in chapter 7 we conclude and discuss about future work.

Chapter 2
Literature Review

In this chapter we review literature related to the insider threat problem, its impact to the confidentiality, integrity and availability of organizational assets, countermeasures and strategies introduced in past and present research. Various research papers review the insider threat problem, present relevant statistics and propose measures in order to mitigate the problem. In the second section of this chapter we present literature focused on computation intelligence inspired by nature.

2.1 Techniques and Measures to Mitigate the Insider Threat

The CERT division, part of Carnegie Mellon University's Software Engineering Institute, provides insider threats' 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 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 (Theis et al., 2019).

Schultz (2002) 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 to detect or predict the insider threat, than using one (Schultz, 2002). 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", stand on attacker's skills and will be hard to detect a skillful attacker.

Salem, Hershkop and Stolfo (2008) conduct a research regarding approaches and techniques for insider threat detection and acknowledge the challenge of building an effective and accurate system for detecting insider attacks.

Brown, Watkins and Greitzer (2013) 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 thread risk (Brown, Watkins and Greitzer, 2013). 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 raised privacy and legal issues.

Axelrad et al. (2013), 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 generates 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 the variables include "personal life stressor and job stressors", personality and capability, attitude, workplace behavior and degree of interest (Axelrad et al., 2013). As the authors acknowledge there are some concerns regarding collection of data for specific variables, such as job satisfaction, in which data may not be accurate.

Nurse et al. (2014) propose a framework to better understand and fully characterize the insider threat problem, developed after analysis of several real-world threat cases and relevant literature. The authors' proposed unifying framework consists 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 times, 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 European Union (EU), the General Data Protection Regulation (GDPR) regulates behavioral observation. Despite regulations, there are many challenges of monitor 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, 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 (Nurse et al., 2014).

Greitzer et al. (2014), propose mitigation strategies and countermeasures for the unintentional insider threat, after their research of regarding cases and papers. The authors review of possible causes and contributing factors and propose measures with an emphasis on employees' continuously training, to recognize threats such as phishing and enhance their awareness on the insider threat problem. Mitigation strategies also include the enforcement of security policies and 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 be become an actual threat (Greitzer et al., 2014).

In order to build mechanisms for detection and prevention of insider threats, real data need to be gathered and this "raise a variety of legal, ethical and business issues" (Glasser and Lindauer, 2013),

Eldardiry et al. (2013), propose a global model approach, based on feature extraction from user activities, logged on large amount of work practice data. This data is comprised of various domain

areas, such as logfiles 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 utilized for easier threat prediction, compared to logon and http history domains (Eldardiry et al., 2013).

Rashid, Agrafiotis and Nurse (2016) 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 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 (Rashid, Agrafiotis and Nurse, 2016).

Lo et al. (2018) apply Hidden Markov Method on CERT's synthetic data set and analyze a number of distance measurement techniques, Damerau–Levenshtein Distance, Cosine Distance, and Jaccard Distance and their performance for detecting changes of user behavior. The authors reports that although HMM outscores each individual distance measurement technique, it needs more than a day to process all data (Lo et al., 2018).

Tuor et al. (2017), propose an online unsupervised deep learning approach to detect anomalies through analysis of the organization’s computer network activity. The authors use Deep and Recurrent Neural Networks (DNN and RNN) and utilizes CERT’s synthetic Dataset to detect anomalous activity.

Le and Zincir-Heywood (2019) 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 that 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 (Le and Zincir-Heywood, 2019).

Liu et al. (2018) 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 area 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 collected 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 deep understanding about the various involved systems. Another challenging problem is extracting the important and relevant features and manage 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 on 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 completely relying on prior domain knowledge (Liu et al., 2018).

2.2 Computation Intelligence inspired by Nature

In this section we present literature which propose models inspired by nature to solve complex problems.

Xiao, Shao and Liu (2006) propose PSO-KM, a K-means algorithm based on Particle Swarm Optimization. The authors acknowledge the challenges of labelling huge amounts of log data and focus their work on detecting unknown attacks automatically, without any prior knowledge on the domain's log data. With the introduction of PSO into the K-means algorithm they present an effective algorithm with the ability of partitioning large datasets and a better global search ability (Xiao, Shao and Liu, 2006).

Del Valle et al. (2008), review the Particle Swarm Optimization (PSO) technique and its application in power system optimization problems and give an insight on how the PSO technique can be used to address complicated engineering optimization problems.

Mohemmed, Zhang and Browne (2010) convert the outlier detection problem into an optimization problem and apply a Particle Swarm Optimization (PSO) based approach, using distance-based measures for outlier detection. The author's approach integrates feature selection ability into their entire framework and directly detects outliers from a particular dataset (Mohemmed, Zhang and Browne, 2010).

Kolias, Kambourakis and Maragoudakis (2011), perform a survey which explores the reasons of the application of Swarm Intelligence (SI) algorithms in the Intrusion Detection field and present various SI methods used for constructing Intrusion Detection Systems (IDS) (Kolias, Kambourakis and Maragoudakis, 2011).

Srinoy (2007), presents an Intrusion Detection model, which uses Particle Swarm Optimization (PSO) for feature selection and Support Vector Machine (SVMs), to detect suspicious activity in the Intrusion Detection domain. The author performs feature selection optimization on an academic widely known dataset in the Intrusion Detection domain, the KDD'99 dataset and employees PSO to find the best optimal feature subset (Srinoy, 2007).

Nakamura et al. (2012) propose a binary version of the Bat algorithm (BAT) to address the problem of high dimensionality. The authors' proposed feature selection technique position the bats in binary coordinates along the search space, which represent a string of bits indicating whether a feature is selected or not after the optimization.

Emary et al. (2016a), propose binary variants of ant lion optimizer (ALO) for wrapper-based feature selection. The author's approaches are applied to find a feature subset in the Machine Learning domain, by minimizing the number of selected features (Emary at al., 2016a).

Emary et al. (2016b) propose two novel binary versions of the grey wolf optimization (GWO) and use it for feature selection optimization, to find optimal subset. The authors propose two binary versions of GWO, which are used to search the feature space for the best combination of features, which has maximum classification accuracy and minimum number of selected features.

Faris et al. (2018) state "Dimensionality is the main challenge that may degrade the performance of the machine learning tasks". The authors acknowledge the challenging problem of searching for the optimal subset in the feature selection optimization process and test the performance of several well-known Swarm Intelligence algorithms in solving this problem.

Khurma et al. (2020), acknowledge the importance of feature selection in the data mining process and propose a Python optimization framework, named "EvoloPy-FS", focused on solving feature selection optimization problems. EvoloPy-FS framework comes with eight nature-inspired metaheuristic optimizers, including particle swarm optimization (PSO), gray wolf optimizer (GWO), bat algorithm (BAT), cuckoo search (CS) and firefly algorithm (FFA), in their binary

presentation. The author's main objective for developing this framework is to help researchers from various domains, with less knowledge in Swarm Intelligence, to setup experiments and get rapid results for their problems, without having to code everything from scratch (Khurma et al., 2020).

2.3 Summary

In the literature reviewed for the purpose of this M.Sc. dissertation, we came up only with a limited study that uses Bio-Inspired computing to mitigate Insider Threats. Much research focus on machine learning based insider threat detection to identify unusual behavior of users in regard with their normal behavior. Machine learning models count on domain knowledge in the feature extraction and selection process, resulting in time consuming during data pre-processing and limited effectiveness on detecting the threats, in cases where domain knowledge is not stated. Several researchers utilized Bio-Inspired models to address optimal solutions in complex problems. We decided to utilize Bio-Inspired computing to enhance Machine learning models, by automating the feature selection process and utilize unsupervised algorithms for outlier detection.

Chapter 3

Swarm Intelligence and Bio-Inspired Computing

In this section we present an overview of Swarm Intelligence and Bio-Inspired computing, to get the reader familiar with the concept and the methods used in this M.Sc. dissertation and we also present the most popular developed algorithms in Bio-Inspired computing.

Bio-Inspired optimization algorithms have emerged to address highly complex problems in science and engineering and provide solutions in time (Kar, 2016; Darwish, 2018). The scientific community, inspired by the biological evolution, has proposed algorithms, such as Neural Network, Genetic Algorithm and Swarm Intelligence, which are just three of the most popular domains, in which meta heuristic optimization methods replicate biological organisms' behavior to address optimization problems (Kar, 2016). The social and foraging behavior of various species in the nature, such as ants, birds, fish, bees, bats and wolves, have attracted the attention of several

researchers to mimic it in and propose algorithms to solve optimization problems in several different fields, including science and engineering (Darwish, 2018).

Swarm Intelligence (SI), an Artifficial Intellience (AI) discipline, studies the collective behavior of nature's species (Li and Clerc, 2018), such as social insects, birds, fishes and other animals. Each individual organization is not considered to be intelligent to solve a problem, but when these species form a swarm and they interact with each other and their environment, they can solve complex problems, such as find the shortest path between their nest and a food source (Saka et al., 2013).

"Swarm intelligence refers to a kind of problem-solving ability that emerges in the interactions of simple information-processing units." (Kennedy, 2006).

A Swarm Intelligence System is composed of a population of individual simple agents and has no centralized control. The interaction between these agents and the interaction of these agents with the environment, creates a problem-solving ability in the SI system and makes the SI system to behave in a complex and self-organized manner to address complex tasks that are too challenging to conventional computation techniques (Li and Clerc, 2019). The processing unit of a swam can be animate, mechanical, computational or mathematical (Kennedy, 2006).

"The concept of a swarm suggests multiplicity, stochasticity, randomness, and messiness, and the concept of intelligence suggests that the problem-solving method is somehow successful." (Kennedy, 2006).

Swarm Intelligence based techniques, such as Ant Colony Optimization (ACO), Particle Swarm Optmization (PSO) and Artificial Bee Colony (ABC), inspired by ant foraging behavior, bird flocking, fish schooling and animal herding, have been applied in the field of intrusion detection (Kolias, Kambourakis and Maragoudakis, 2011).

Bio-Inspired algorithms have been used in much research to address several real-world challenging problems and portion of this research used more than a single algorithm to address the same problem, to compare the performance of the algorithms on that problem's domain. Literature reports that not all algorithms perform the same in solving specific set of problems and researchers experiment with modifications of existing algorithms or even propose new algorithms to reach a better performance in their domain of research (Mirjalili, 2015). **Table 1** displays bio-inspired

algorithms, as presented by the related literature reviewed for this M.Sc. dissertation, with the scope of applications in which they have been used.

Algorithms	Scope of Application
Neural Networks (NN)	Association rules, pattern classification, regression, feature selection, missing data prediction, sequence mining, data reduction, probabilistic prediction, Bayesian and deep learning, feature detection, speech and image recognition, synchronization, control of non-linear systems, switching networks (Kar, 2016).
Genetic Algorithm (GA)	Search, maximization or minimization, sorting, multi-criteria selection, job allocation, process scheduling, network analysis, anomaly detection, intrusion detection, parallel computation, prioritization, classification, network path routing, load balancing problems, layout planning, signal coordination, sorting, structural systems (Kar, 2016).
Particle Swarm Optimization (PSO)	Distributed resource management, search, location identification, resource allocation, regulation, chaotic systems, oscillatory systems, global optimization, path optimization, adaptive learning, job scheduling, thresh-holding, network training, minimization, maximization, migration (Kar, 2016).
Ant Colony Optimization (ACO)	Network analysis, travelling salesman problem, scheduling, routing, clustering, data compression, environmental and economic dispatch problems, routing, data reconciliation, parameter estimation, gaming theory, objective tracking, demand forecasting, layout design, continuous optimization, timing optimization, resource consumption optimization (Kar, 2016).

Artificial Bee Colony (ABC)	Numerical function optimization, multilevel thresh-holding, network routing, allocation / assignment, test suite optimization, search, bench-marking, probability distribution, feature selection, single and multi-objective optimization, discrete and continuous optimization (Kar, 2016).
Bacterial Foraging Optimization Algorithm (BFOA)	Multi-optimal function optimization, numerical optimization, global optimization, gradient based search, linear combiners, forecasting models, minimization and maximization, load forecasting and compensation, portfolio value prediction, clustering, load dispatch and calibration (Kar, 2016).
Cuckoo search (CS)	Search problems, multi-objective problems, optimization among designs, gradient based optimization, gradient free optimization, multi-objective scheduling, multi-objective allocation, phase equilibrium problems, reliability optimization, path identification for network analysis, knapsack problems (Kar, 2016).
Firefly Algorithm (FA)	Digital Image Compression, Feature Selection (Yang and He, 2013a). Dispatch problems, job scheduling, chaotic problems, structural optimization, continuous optimization, vector quantization, clustering, price forecasting, discrete optimization, load forecasting, network analysis, travelling salesman problem, non-linear optimization, dynamic environment problems (Kar, 2016).
Leaping Frog Algorithm	Optimization problems from structures, mechanical systems, neural networks, chemistry (Snyman, 2000). Combinatorial problems, network design problems, job scheduling problems, thresh-holding problems, network scaling problems, cost minimization problems, permutation

	based searching problems and resource constrained problems (Kar, 2016).
Bat Algorithm (BA)	Structural design optimization, multi-objective optimization, numerical optimization problems, network path analysis, multi-constrained operations, adaptive learning problems, environmental/economic dispatch, scheduling, effort estimation, classification, vector matching and association rule mining (Kar, 2016).
Flower Pollination Algorithm (FPA)	Control in multi-machine systems, feature selection, structure optimization, data reduction, array synthesis, classification, search, multi-criteria selection, chaotic systems, electro-magnetics, large scale linear programming, energy management, structural engineering (Kar, 2016).
Artificial Plant Optimization Algorithm (APOA)	Protein analysis, network configuration simulation analysis, coverage optimization, telecom sensor networks, molecular structure analysis (Kar, 2016).
Ant Lion Optimizer (ALO) – imitates the hunting process of ant lions	Feature selection (Emary et al., 2016a),
Grey Wolf Optimizer (GWO)	Feature selection (Ematy et al., 2016b), attribute reduction strategy, route planning, reduced parametric sensitivity (Darwish, 2018)

Table 1: Bio-Inspired algorithms

3.1 Neural Networks (NN)

Neural Networks (NN) have been widely applied to solve problems in the field of pattern recognition, image recognition, speech recognition and natural language processing (Garro, Sossa

and Vazquez, 2009). Artificial Neural Networks attempt to simulate the networks of neurons of an intelligent organism, such as the nerve cells of a human's brain, by combining multiple processing units, the neurons, into a self-adapting and self-organizing system (Sarle, 1994). These systems can learn to perform various tasks, such as classify or recognize patterns, based on inputs and feedback from each node in the neural network (Kar, 2016). There are many approaches for implementing neural networks, with the simplest one being the Perceptron networks, which can be used for both linear and non-linear systems. Deep Neural Network (DNN) has been utilized by Yuan et al. (2018) in their proposed insider threat detection method, in which they use Long Short Term Memory (LSTM) to "learn" the user behavior and extract abstracted temporal features, which are converted to fixed-size feature matrices to be used by the Convolutional Neural Network (CNN) to detect insider threat. Yuan etl a. (2018) reviews literature that explores various implementations of Neural Networks, including Recurrent Neural Networks (RNN), to detect anomalous behavior. An extensive literature exists, reporting approaches that utilize Neural Networks (NN) in several subject areas, including Engineering, Computer Science, Mathematics, Physics and Astronomy (Kar, 2016). Garro, Sossa and Vasquez (2009) acknowledge that "neural networks cannot reach an optimum performance in non-linear problems" and utilize Particle Swarm Optimization (PSO), to automatically design an Artificial Neural Network (ANN). PSO is used to find all relevant values and functions and optimize the error function.

3.2 Genetic Algorithm (GA)

Genetic Algorithm (GA), a population-based Evolutionary Algorithm (EA), attempts to simulate the phenomenon of natural evolution and natural genetics, where the fittest individuals are selected for reproduction, to identify good and working solutions (Kar, 2016). The literature reports several versions of Genetic Algorithm implementations (Hassan, Cohanim, de Weck and Venter, 2005). The implementation of GA starts with a set of individuals, the population, with everyone representing a solution to the problem. GA employs a fitness function to determine the fittest individual and computes a fitness score for each individual. The generation is represented by every iteration and over several generations, the **selection** operator, which represents the "survival of the fittest" principal, is employed to select the fittest individuals with the most positive characteristics, based on its fitness score. The **crossover** operator is then used to produce new solutions based on the positive characteristics of the current population, to propagate them to the future population. The **mutation** operator is employed to ensure diversity within the new population and avoid local optimality (Hassan, Cohanim, de Weck and Venter, 2005; Kar, 2016).

3.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), introduced and developed by Kennedy and Eberhart in 1995, is a population-based optimization and meta-heuristic technique, inspired by the social behavior of bird flocking, fish schooling and swarming theory (Kennedy and Eberhart, 1995). PSO is considered by researchers and practitioners an efficient and effective technique for solving complex optimization problems (Li and Clerc, 2019). PSO is composed of a swarm of particles and each individual particle represents a candidate solution. The particles move into the problem search space with a certain velocity and their movement is affected by their own best position found so far, which is defined as the **pbest** quality factor and the global best solution found by their neighbors in the search space, which is defined as the **gbest** quality factor (Kennedy and Eberhart, 1995). The position of each particle or the quality of the solution, is evaluated by the fitness function. Each particle tries to modify its position using the equations (Zhan et al., 2009; Kolias, Kambourakis and Maragoudakis, 2011):

1. $$\boldsymbol{vi(t+1) = \omega \cdot vi(t) + c1 \cdot r1 \cdot (pBesti - xi) + c2 \cdot r2 \cdot (gBest - xi)} \quad (1)$$

2. $$\boldsymbol{xi\,(t+1) = xi + vi\,(t+1)} \quad (2)$$

Equation (1) updates the particle's speed and equation (2) updates the particle's position. In (1) vi is the particle's speed, ω is the inertia weight constant, c1 and c2 are the acceleration coefficients, r1 and r2 are random numbers generated within [0,1], pBesti is the particle's position with the best fitness found so far, gBest is the swarm's global best position and xi is the particle's current position (Kolias, Kambourakis and Maragoudakis, 2011). The particle's velocity and position are initialized randomly and then are updated using the equations (1) and (2).

"Particle Swarm Optimization is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions" (Kennedy and Eberhart, 1995).

3.4 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a population-based metaheuristic technique, used for solving combinatorial optimization problems. ACO was introduced by M. Dorigo and his colleagues, in early 1990s, inspired by the behavior of real ants (Dorigo and Di Caro, 1999; Dorigo and Blum, 2005).

Even though a single ant has limited intelligence and capabilities, when an ant works together with other ants in an ant colony, they can be very well organized and effective to perform tasks that requires intelligence, such as transport heavy goods or finding the shortest path between a food source and their nest. Ants communicate with each other through a chemical substance, known as pheromone, which they generate to relay a message, such as to give directions to other ants in their colony (Chakraborty and Kar, 2017). An ant evaluates the quality and quantity of a food source as soon as it finds one and generates pheromone during its way back the nest. The quality of the pheromone generated on the ant's trail, depends on the quality and quantity of the food source. Ants will follow the pathway created by the pheromone and will renew the trail, depending of the value of the source (Dorigo and Blum, 2005). Pheromone on the shortest path will be reinforced sooner than the other paths, so the shortest path will have greater pheromone concentration and higher probability of being selected from other ants (Kolias, Kambourakis and Maragoudakis, 2011). **Figure 1** shows the random path selection (a) by ants in the search for a food source. The shortest path is reinforced with more pheromone and attracts more ants and eventually the pheromone trails on longer paths evaporate over time (b).

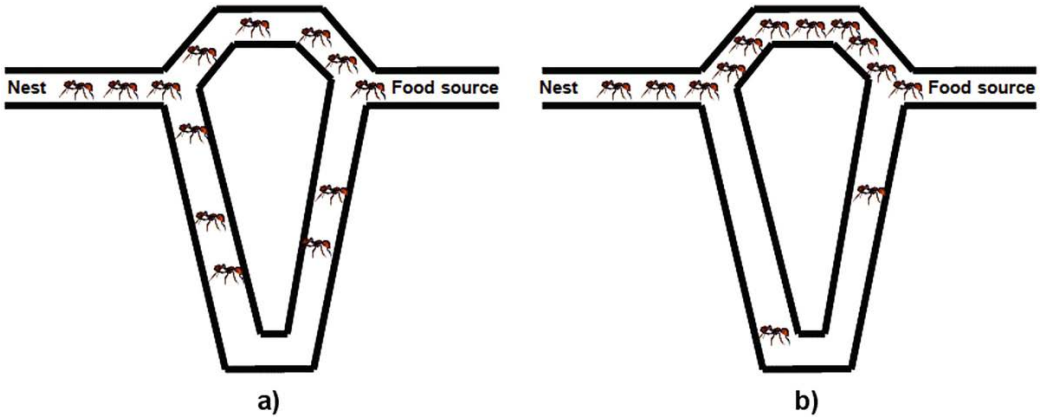

Figure 1: Ants seek for the shorter path of the two, between their nest and the food source (Li and Clerk, 2019).

3.5 Artificial Bee Colony (ABC)

Artificial Bee Colony (ABC) is a bio-inspired algorithm, which mimics the behavior of swarm intelligence of bees in the way they communicate, navigating, selecting their nest, mating and floral foraging, to search for an optimal numerical solution among a large number of candidates (Karaboga and Basturk, 2007; Kar, 2016). ABC algorithms depends on the solution and the quality of the solution, the fitness. In the ABC algorithm the colony consists of three groups, the employed

bees, the onlookers and the scouts. The random search for food sources is carried out by the scouts, with the onlookers waiting to make the decision about the food source (Karaboga and Basturk, 2007). A candidate solution is the representation of a source of food, which is identified by the employed bees and the quality of the solution, is the representation of the amount of nectar in that source. For every newly discovered source of food, its fitness is computed to decide whether to adopt or reject this new source, according to its fitness (Ghanem and Jantan, 2014; Kar, 2016). The main steps of the algorithm are (Karaboga and Basturk, 2007):

- Intitialize
- REPEAT UNTIL [Requirements are met]
    - Place the employee bees in the memory
    - Place the onlooker bees in the memory
    - Send the scouts to the search area to discover new food sources

3.6 Bacterial Foraging Optimization Algorithm (BFOA)

Bacterial foraging optimization algorithm (BFOA) is a global optimization algorithm, based on the social foraging behavior of Es-cherichia coli bacteria. Bacteria search for food in a way that their energy intake is maximized per unit time. The process in which an Individual Bacterium is searching for food is called **chemotaxis**, which is the idea behind BFOA (Das et al., 2009). In BFOA, an agent representing the bacterium, searches for a local suitable solution. The agent utilizes the operators of chemotaxis, swarming, reproduction and elimination-dispersal to locate the global optimum (Das et al., 2009; Kar, 2016). Das et al., (2009) and Kar (2016) report that while the algorithm is easy to implement, it has a poor convergence capability for solving complex optimization problems.

3.7 Cuckoo Search (CS)

Cuckoo Search (CS) mimics the breeding behavior of the cuckoo bird species. Cuckoos have a very interesting and aggressive reproduction strategy. Instead of laying their eggs in their own nests, they lay them on communal nests and sometimes they even remove the foreigners' nest eggs to increase the hatching probability their own eggs (Gandomi et al., 2011a). In the CS algorithm, cuckoos are represented by the agents and the eggs are represented by the candidate solutions. A cuckoo lays one egg at a time and dumps it in a random nest. The nests are discovered through Levy

flights. The best net, with the higher quality of eggs (optimal candidate solutions) will carry over to the next generations. The number of available host nests is fixed and there is a probability that the cuckoo's egg is discovered by the host bird. In this case the host bird will either throw away the foreign egg, or it will abandon the nest and build another nest, which will become a future potential nest for the cuckoo (Gandomi et al., 2011a; Kar, 2016).

3.8 Firefly Algorithm (FA)

Firefly Algorithm (FA) mimics the social behavior of fireflies' flashing characteristics. Fireflies use flashing patterns to communicate, find mates or search for prey. FA follows three rules (Yang, 2010b; Gandomi et al., 2011b; Yang and He, 2013a):

- Fireflies are unisex and are attracted to each other, regardless of their sex
- Attractiveness is proportional to their brightness. A less bright firefly will move towards a brighter one and if there isn't a brighter one it will move randomly in space.
- The firefly's brightness is related with the objective of the optimization problem.

The movement of a firefly I, attracted by a firefly j is computed using the equation (3) (Emary et al., 2015).

3. $$\boldsymbol{x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(rand - 0.5)} \quad (3)$$

In equation (3), x represents the movement of a firefly i, which is attracted by a firefly j. α is the randomization parameter, rand is a random number in [0,1] and (rand – 0.5) represents number in [-0.5, 0.5], since variation can be positive or negative. $\boldsymbol{\beta_0}$ is set to 1 and α is in [0,1] (Emary et al., 2015).

Gandomi et al., (2011b) acknowledge the efficiency of FA, but also reports observations of oscillatory behavior as the search process approaches the optimum design. The authors suggest a gradually reduction of the randomization parameter as the optimization processes, to improve the algorithm (Gandomi et al., 2011b).

3.9 Leaping Frog Algorithm

Leaping Frog Algorithm is based on frog foraging behavior and was developed to solve unconstrained optimization problems but has been also used to solve constrained problems as well (Snyman, 2000). The popular extension of the algorithm, Shaffled Frog-Leaping Algorithm (SFLA), is a metaheuristic algorithm which combines the benefits of genetic-based memetic algorithm and social behavior-based swarm optimization algorithms (Eusuff, Lansey and Pasha, 2006; Fang and Wang, 2012). In SFLA an initial population of randomly generated virtual frogs, who represent candidate solutions. is formed and then the population is partitioned into subsets, called memeplexes. A subset of the memeplex is then constructed for each initial subset, based on probability distributions. For each of the subsets of the memeplex, called sub-memeplex, the worst frog will leap towards a food source, based on its own experience but also the experience from the best frog of that sub-memeplex. If the new position is better than the old one, the frog repeats the process, else the worst frog is replaced by a new randomly generated frog (Fang and Wang, 2012).

3.10 Bat Algorithm (BA)

Bat Algorithm, proposed by Yang (2010a) for solving optimization problems, mimics the behavior of bats during the search for a prey or food, using their advanced capability of echolocation. Echolocation is a type of sonar, used by bats to detect prey by avoiding obstacles and to locate their roost. Each bat **i** flies with a velocity **vi** at position **xi**, representing the solution and use a fixed frequency of **fmin**, a varying wavelength **λ** and loudness **A0** to search for prey. During the simulation of the bats' movement in a d-dimensional search space, the position and velocity are calculated using the equations (4), (5) and (6) (Yang, 2010a).

4. $$f_i = fmin + (fmax - fmin) \,.\, \beta \quad (4)$$

5. $$vi^t = vi^{t-1} + (xi^t - x_*).f_i \quad (5)$$

6. $$xi^t = xi^{t-1} + vi^t \quad (6)$$

In equations (4), (5) β is a random vector in [0,1] and x_* represents the current global best solution among all virtual bats.

3.11 Flower Pollination Algorithm (FPA)

Flower Pollination Algorithm (FPA) is based on the pollination process of flower plants and was developed to solve global optimization problems. During the pollination process, the agents, named pollinators who represent the insects, wind and water, are employed to spread the flowers' pollens to another plant for reproduction. Pollination can be achieved by self-pollination, which represents local optimization, in which process the involved flowers are from the same plant, or by cross pollination, which represents global optimization, in which process the involved flowers are from different plants. For simplicity the algorithm assumes that each plant has only one flower and each flower produce a single pollen (Yang, 2012; Yang, Karamanoglu and He, 2013). Yang (2012) acknowledges the efficiency of FPA due to long-distance pollinators and flower consistency.

3.12 Artificial Plant Optimization Algorithm (APOA)

Artificial Plant Optimization Algorithm (APOA), inspired by tree's growing process, was developed to address global optimization problems. The algorithm simulates the plant growing phenomenon, by connecting the growing process with an optimization problem. Each new iteration represents the plant's growing period, global optimum represents the highest light intensity, fitness value represents the light intensity, a point represents a branch and position update represents branch growth. Each branch of the plant represents a potential solution to the optimization problem and its fitness is calculated during the simulated process of photosynthetsis (Cui and Cai, 2013).

3.13 Ant Lion Optimizer (ALO)

Ant Lion Optimizer (ALO) is a bio-inspired optimization technique, inspired by the hunting mechanism of ant lions in nature, developed to solve optimization problems. Antlions create small cone-shaped traps in sand to catch their preys. After the creation of their traps they hide at the bottom of the cone and wait for their preys to enter the trap. Once the prey is in the trap, the antlion tries to catch it. It has been observed that the size of the trap is relevant with the level of hunger and shape of the moon. Antlions tend to dig deeper traps when they are hungry. This foraging behavior is the main inspiration of the ALO algorithm. ALO simulates five main steps of the antlions' hunt behavior (Mirjalili, 2015):

- Random walk of ants
- Building traps
- Entrapment of ants in traps
- Catching preys
- Re-building traps

3.14 Grey Wolf Optimizer (GWO)

Grey wolf Optimizer (GWO) is a Bio-Inspired optimization technique, which simulates the leadership hierarchy and hunting behavior of grey wolves in nature. Grey wolves travel in packs of five to twelve on average and they follow a strict social dominant hierarchy. The pack is led by the alphas, a male and a female, who are making decisions such as time of hunting, deciding resting place, etc. The betas, best candidates to be alphas, are next in rank and assist the alphas in the decision-making process. The deltas follow orders from alphas and betas but dominate the omegas who are last in the rank and follow orders of all other dominant wolves. In GWO social hierarchy's mathematical model, the fittest solution is called the alpha (α), with the second and third best solutions named beta (β) and delta (δ), respectively. All the rest candidate solutions are assumed to be omega (ω). The GWO hunting mathematical model is comprised of tracking, encircling and attacking the prey. Hunting of a prey starts with the pack encircling. The equations (7), (8), (9), (10) are used to mathematically model grey wolves' encircling behavior (Mirjalili, Mirjalili and Lewis, 2014).

7. $$\boldsymbol{D} = | \boldsymbol{C} \,.\, \boldsymbol{Xp}(\boldsymbol{t}) - \boldsymbol{X}(\boldsymbol{t}) |$$ (7)

8. $$\boldsymbol{X}(\boldsymbol{t}+\boldsymbol{1}) = \boldsymbol{Xp}(\boldsymbol{t}) - \boldsymbol{A} \,.\, \boldsymbol{D}$$ (8)

9. $$\boldsymbol{A} = \boldsymbol{2a} \,.\, \boldsymbol{r1} - \boldsymbol{a}$$ (9)

10. $$\boldsymbol{C} = \boldsymbol{2} \,.\, \boldsymbol{r2}$$ (10)

In Equations (7) and (8), **t** represents current iteration and A, C are the coefficient vectors. D is the distance factor, Xp represents the position of the prey and X represents the position of a grey wolf. A and C are calculated in equations (9), (10), where **a** is linearly decreased from 2 to 0 through the number of iterations and **r1, r2** are random vectors in [0,1] (Mirjalili, Mirjalili and Lewis, 2014). The alphas guide the hunting process with the participation of betas and deltas. The alpha, beta and delta, have better knowledge about the prey's location and they represent the best three candidate solutions. The rest search agents, including the omegas update their position according to the best search agents. The equation (11) is used to update wolves' position.

$$\mathbf{11.}\ \boldsymbol{X(t+1)} = \frac{\boldsymbol{X1+X2+X3}}{\boldsymbol{3}} \tag{11}$$

$$12.\ \boldsymbol{X1} = |\boldsymbol{X\alpha} - \boldsymbol{A1}\ .\ \boldsymbol{D\alpha}| \tag{12}$$

$$13.\ \boldsymbol{X2} = |\boldsymbol{X\beta} - \boldsymbol{A2}\ .\ \boldsymbol{D\beta}| \tag{13}$$

$$14.\ \boldsymbol{X3} = |\boldsymbol{X\delta} - \boldsymbol{A3}\ .\ \boldsymbol{D\delta}| \tag{14}$$

$$15.\ \boldsymbol{D\alpha} = |\boldsymbol{C1}\ .\ \boldsymbol{X\alpha} - \boldsymbol{X}| \tag{15}$$

$$16.\ \boldsymbol{D\beta} = |\boldsymbol{C2}\ .\ \boldsymbol{X\beta} - \boldsymbol{X}| \tag{16}$$

$$17.\ \boldsymbol{D\delta} = |\boldsymbol{C3}\ .\ \boldsymbol{X\delta} - \boldsymbol{X}| \tag{17}$$

The parameters X1, X2 and X3 in equation (11), are defined in equations (12), (13) and (14), respectively, where Xα, Xβ and Xδ represent the first three best candidate solutions in the swarm at a given iteration A1, A2 and A3 respectively. Dα, Dβ and Dδ are defined in equations (15), (16) and (17), respectively.

The attack or exploitation phase of the prey comes when the prey stops moving. The mathematical representation of the grey wolf approaching the prey, is done by decreasing the value of **a** over the course of iterations. By decreasing a, A calculated by equation (9) is also decreased.

3.15 Comparison of Bio-Inspired Models

Not all Bio-Inspired algorithms have similar performance on a specific real-world problem. Furthermore, each algorithm has its advantages and disadvantages as reported in literature and presented in **Table 2**.

Algorithm	Advantages	Disadvantages
Neural Networks (NN)	Can be combined with other algorithms to improve the predicting capabilities of the system (Kar, 2016)	Cannot reach an optimum performance in non-linear problems (Garro, Sossa and Vazquez, 2009)
Genetic Algorithm (GA)	Easy of operations, minimal requirements, global perspective (Sivanandam and Deepa, 2008). Efficiency, extraordinary robustness regarding input data (Meyer-Baese and Schmid, 2014). Intuitiveness, ease of implementation (Hassan, Cohanim, de Weck and Venter, 2005).	Reduced performance when applied to very complex and high dimensional problems (Kar, 2016). Poor local search ability (Karaboga and Basturk, 2007). A precise knowledge of the basics and the context is crucial for any problem solution (Meyer-Baese and Schmid, 2014). Expensive computation cost (Hassan, Cohanim, de Weck and Venter, 2005), have drawbacks in dealing with multimodal optimization problems (Yang and He, 2013b)

Particle Swarm Optimization (PSO)	Extremely simple algorithm, effective on optimizing a wide range of functions (Kennedy and Eberhart, 1995). Effectively on solving large-scale nonlinear optimization problems (del Valle et al., 2008). Calculation of PSO is simple (Bai, 2010)	Suffers from partial optimism (Bai, 2010), have drawbacks in dealing with multimodal optimization problems (Yang and He, 2013b)
Ant Colony Optimization (ACO)	Strong global search ability, can be easily combined with other algorithms, strong robustness (Wang, 2018)	Large number of parameters, slow convergence speed (Wang, 2018)
Artificial Bee Colony (ABC)	Easy to understand and implement (Ghanem and Jantan, 2014). Strong robustness, fast convergence, high flexibility, few parameters to configure (Yan, 2011)	Premature convergence (Yan, 2011)
Bacterial Foraging Optimization Algorithm (BFOA)	Easy to implement (Das et al., 2009; Kar, 2016).	Poor convergence capability for solving complex optimization problems. (Das et al., 2009; Kar, 2016).
Cuckoo search	Few parameters to be fine-tuned (Gandomi et al., 2011a).	Slow convergence (Wang, 2018)

	Strong global search ability (Wang, 2018)	Local search is not careful (Wang and Li, 2019)
Firefly Algorithm	Automatically subdivision, ability to deal with multimodality (Yang and He, 2013a)	Oscillatory behavior as the search process approaches the optimum design (Gandomi et al., 2011b)
Leaping Frog Algorithm	High search precision (Wang and Li, 2019)	Slow convergence velocity, falling easily to local optimum (Wang and Li, 2019)
Bat Algorithm	Simplicity and flexibility, very quick convergence (Yang and He, 2013b)	Convergence slows down after early stage (Yang and He, 2013b)
Flower Pollination Algorithm (FPA)	Explores large search space (Yang, 2012). Simplicity and Flexibility (Yang, Karamanoglu and He, 2013)	Premature convergence, poor exploitation ability (Cui and He, 2018)
Artificial plant optimization	Self-organization, self-learning (Cui and Cai, 2013)	New algorithm – further exploration and research is needed (Kar, 2016)
Ant Lion Optimizer (ALO)	High exploitation and convergence rate (Mirjalili, 2015)	Long run time due to the random walking process Kilic, Yuzgec and Karakuzu, 2018

Grey Wolf Optimizer (GWO)	Easy to implement, Faster convergence, Avoidance of local optima, high performance in unknown challenging search spaces (Mirjalili, Mirjalili and Lewis, 2014). Low computational cost (Darwish, 2018), high search precision (Wang and Li, 2019)	New algorithm – further exploration and research is needed (Wang and Li, 2019)

Table 2: Bio-Inspired algorithms comparison table

3.16 Selection of Bio-Inspired Models

For the purpose of this M.Sc. dissertation we decided to employ three algorithms and test their performance on feature selection optimization. Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) are similar population-based search algorithms, but according to Hassan, Cohanim, de Weck and Venter (2005) PSO has superior computation efficiency compared to GA. PSO, GA, ACO and NN have been implemented in many studies in literature (Kar, 2016), so we decided to choose only one of this group of algorithms and compare it with algorithms that have not been explored on a similar level. Khurma et al. (2020) reports that the binary version of PSO, BPSO, has the highest stability in error rate results on feature selection optimization and that makes it a perfect candidate algorithm for our approach. Khurma et al. (2020) also reports that BBAT optimizer, the binary version of Bat Algorithm, was the fastest optimizer on all the authors' tests, compared with all the other optimizers. Kar (2016) suggests the application of Bat algorithm for further exploration, so we decided to use this algorithm as well. Lastly, we chose an algorithm from the "zone of theory development" (Kar, 2016), GWO, that reports high performance with low computation cost.

3.17 Summary

In this chapter we presented an overview of Swarm Intelligence and Bio-Inspired computing and presented the most popular and well-known algorithms, such as Neural Networks, Genetic Algorithm, PSO and ACO, but also more recent developed algorithms in Bio-Inspired computing, such as BA, GWO, FA and CS. We presented advantages and disadvantages of each algorithm as reported in literature and selected three (3) Bio-Inspired models to utilize in our approach.

.

Chapter 4
Methodology

In chapter 2 we reviewed literature focused on machine learning based insider threat detection and the need to improve the performance of such models. In this chapter we present the methodology on building a Machine Learning model-based insider threat detection and improving its performance by utilizing Swarm Intelligence algorithms.

4.1 Overview

In real-world environments, a security system is in place to collect and store log data for further processing. Such system is a Security Information and Evet Monitoring (SIEM) system, designed to collect log data from various type of systems, such as domain controllers, web and application servers, databases, network and security devices, email systems, etc. Since each device uses a different format for their log data, log parsing is mandatory after the collection of log data. The logs are then combined in a way that data related to a specific security incident can be correlated, to help a security analyst to escalate the incident. Log data contain an enormous number of features, such

as user id, computer name, ip address, mac address, data, time and many features related with the source device, such as policy id, service information, action, etc. and many of these features are irrelevant with security incidents and are just creating noise. Feature selection optimization comes a solution to this kind of problems.

Our insider threat detection 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 omit unneeded or irrelevant and redundant features, to improve a model's prediction performance, reduce resource requirements, processing and utilization times (Guyon and Elisseeff, 2003).

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 our algorithms can be more efficient with. An automatic feature selection optimization, which is described in section 4.3, is then applied on the generated data frame to produce the optimum subset of features. In **Figure 2** we illustrate the components of our proposed system and the flow of data between them.

Since we are measuring the performance of three (3) different Swarm Intelligence algorithms for feature selection optimization, we now have several results of subset selections to fit into the Machine learning model, described in section 4.4, for detecting insider threats.

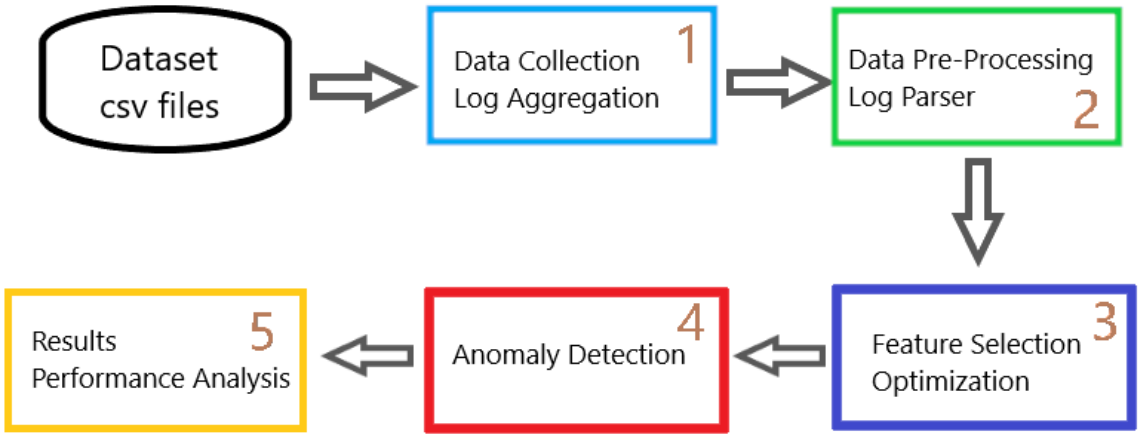

Figure 2: Diagram of proposed method

4.2 Dataset

Data is an essential element in threat detection models and plays a crucial role on the detection of security incidents (Sun et al., 2019). In order to avoid legal and privacy issues we are using a publicly available synthetic dataset. The synthetic datasets contain logs generated specific for insider threat research and each data set contains a small number of insider threat incidents in regard with the dataset's accompanied scenario (Glasser and Lindauer, 2013).

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.

4.3 Feature Selection Optimization using Swarm Intelligence Algorithms

For the purpose of this M.Sc. dissertation, we decided to use EvoloPy-FS framework and measure the performance of three (3) popular Swarm Intelligence algorithms on the feature selection optimization problem.

EvoloPy-FS, an easy to use Python framework, developed by its authors to researchers in solving optimization problems using Swarm intelligence algorithms. The main component of the framework is the Optimizer, where we setup our experiment along with the initial configurations. In the optimizer we define the dataset to use, the optimizers, 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 (Faris et al., 2016; Aljarah et al., 2018; Faris et al., 2018; Mafarja et al., 2018; Faris et al., 2020; Khurma et al., 2020).

In our approach, for feature selection optimization we selected Binary Particle Swarm Optimization (BPSO), Binary Gray Wolf Optimizer (BGWO) and Binary Bat Algorithm (BBAT), as optimizers. The source for obtaining the source code of the framework is presented in Appendix A, section A.2.

4.4 Utilizing Machine Learning for Outlier detection

As stated in section 1.3, Machine learning is applied in several applications and one of them is anomaly detection. Several approaches, reported in the literature, utilize Machine learning as an effective method for detecting anomalies. For the purpose of this M.Sc. dissertation we developed a Machine learning system, focused on user-centered 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 data frame, based on the selected features and not based on CERT's accompanied scenarios. The Machine learning system is presented in full detail in chapter 5.

Local Outlier Factor (LOF), is an unsupervised anomaly detection algorithm, which uses a score to determine if a certain point is an anomaly. Each data point is assigned this score, which is the result of the computation of local density deviation of the given data point with respect to its neighbor data points. If a given data point has a substantially lower density than its neighbors, then it is a considered as an outlier (Breunig et al., 2000). LOF algorithm is considered as an efficient method to detect outliers in high dimensional datasets. In our system we employed sklearn.neighbors.LocalOutlierFactor from scikit-learn module, which utilizes K-nearest neighbours to compute the local density of a data point. LOF score value is determined from the ratio of the average local density of the observation's neighbors and its own local density (Pedregosa et al., 2011)

4.5 Technology and Libraries

For the purpose of this M.Sc. dissertation we used and utilized Python programming language along with several open source libraries. Python was installed with the use of Anaconda, the most popular Python distribution.

4.5.1 Anaconda

Anaconda is an open source Python distribution used to perform data science and build machine learning models (Anaconda Software Distribution, 2019). We used Anaconda to automatically install Python and several other libraries including Pandas, Jupyter and scikit-learn.

4.5.2 Python

Python is a high-level platform independent programming language, quick and easy to learn. Python is the most popular programming language in the Machine learning domain and can be used to create and run a Machine learning model on a single machine (Python Software Foundation. 2019). All processing steps of our system are implemented in Python.

4.5.3 Jupyter

Jupyter notebook is an open source interactive computing notebook environment which supports over 40 programming languages, including Python. Jupyter is very popular in the data science domain, since it combines code, narrative text and visualizations (Project Jupyter, 2020). Jupyter was used to develop the proposed Insider Threat detection model in this M.Sc. dissertation, due to its capability of reproducing research results and the flexibility of using multiple cells and running slices of code, which makes it easy to work with data analysis.

4.5.4 Pandas

Pandas is a powerful and high-performance data analysis and manipulation tool, developed for Python, and used for working with large data frames. With Pandas we can easily read data from a variety of formats, including csv (McKinney, 2010; The pandas development team, 2020).

4.5.5 Numpy

Numpy is a fundamental Python library which adds support for multi-dimensional arrays and matrices, along with high-level mathematical functions (Oliphant, 2006).

4.5.6 EvoloPy-FS

As stated in the section 4.3, EvoloPy-FS is Python optimization framework that includes several swarm intelligence algorithms and provide solutions for feature selection optimization problems.

4.5.7 Scikit-learn

Scikit-learn is an open source machine learning Python module, which integrates supervised and unsupervised algorithms into Python for building machine learning models (Pedregosa et al., 2011). The Machine Learning algorithms are employed by utilizing scikit-learn module.

4.6 Performance Metrics

In order to measure the performance of the subset dataset, generated after feature selection optimization, we followed Ferreira, C. Le and Zincir-Heywood's (2019) performance metrics' method, who used insider detection rate and insider detection precision. As mentioned in section 4.4, 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 produce the best precision will be selected as the optimal one. Insider detection rate (DR) and precision can be determined by using the equations (17) and (18) (Ferreira, C. Le and Zincir-Heywood, 2019):

18. $$\boldsymbol{DR} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (17)$$

19. $$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad (18)$$

In equations (17) and (18) TP, short for True Positive, represents the number of true malicious users detected, FP, short for False Positive, represents the number of normal users that were detected as malicious and FN, short for False Negative, represents the malicious users that were not detected as malicious, but falsely considered as normal users.

4.7 Summary

In this chapter we presented the methodology we used to undertake our research and build a Machine Learning model-based insider threat detection, while improving its performance by utilizing Swarm Intelligence algorithms.

Chapter 5
Machine Learning Model

In this chapter we present the Machine Learning Model that was implemented and used for the purpose of this M.Sc. dissertation. Our system is focused on Feature Selection and detects malicious insiders by utilizing unsupervised learning method.

5.1 System Overview

In **Figure 2** we illustrate an overview of our proposed system for malicious behavior detection. In the first step we collect the data from the various sources as well 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 is performed to generate the optimal subset. In the fourth step we fit the anomaly detection algorithm into the generated subset to detect outliers. In the final step we analyze results and measure the algorithms performance.

5.2 Data Collection and Pre-Processing

For the evaluation of our system we have used publicly available synthetic dataset that has no privacy constrains or legal issues.

5.2.1 Dataset Overview

The synthetic dataset that we evaluated in our Machine Learning Model 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" (Glasser and Lindauer, 2013) to allow researchers to experiment with it and evaluate algorithms. There are various datasets to choose from, with most of them having once 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 scenarios with multiple users involved in each scenario. The r4.2 dataset is split up in seven (7) different parts as shown in **Table 3.** 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. Psycometric.csv provides personality scores for each user, based on big five personality traits. Since the used dataset is a synthetic one and there are no privacy or legal constraints on using this dataset's personality scores, we could add these features as well to experiment with. On the contrary, 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 is 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.

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	Users' psychometric scores based on big five personality traits
ldap	A set of eighteen (18) files regarding the organizations' users and their roles

Table 3: Insider Threat Dataset r4.2 files' description

5.2.2 Scenarios

The Insider threat dataset r4.2 contains three (3) attack scenarios (Glasser and Lindauer, 2013):

1. A user who did not previously use removable drives or work after hours, begins logging in after hours, using a removable drive and uploading data to wikileaks.org. The same user leaves the organization shortly thereafter.

2. A use begins surfing job websites and soliciting employment from a competitor. Before leaving the company, the user uses a thumb drive to steal data.

3. A system administrator becomes disgruntled, downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day he uses the collected credentials to log in as his supervisor and send out an alarming mass email causing panic in the organization. He leaves the organization immediately after this event.

5.2.3 Log Files Structure

Table 4 shows the log file structure for each file component of the dataset. As mentioned, in section 5.2.1, we are not going to collect data from "Psychometric.csv" file and we are also dropping other

unnecessary features, to avoid overhead processing and develop a simple to use model. Furthermore, we followed Rashid, Agrafiotis and Nurse (2016) approach and chose features that can be used to model user behavior across several different domains.

File	Column	Datatype
Device.csv	Id	Object
	Date	Object
	User	Object
	Pc	Object
	activity	Object
Email.csv	Id	Object
	Date	Object
	User	Object
	Pc	Object
	To	Object
	Cc	Object
	Bcc	Object
	From	Object
	Size	Int64
	Attachments	Int64

	Content	Object
File.csv	Id Date User Pc Filename Content	Object Object Object Object Object Object
http.csv	Id Date User Pc url content	Object Object Object Object Object Object
Logon.csv	Id Date User Pc Activity	Object Object Object Object Object

Psychometric.csv	Employee_name User_id O C E A N	Object Object Int64 Int64 Int64 Int64 Int64
LDAP (files)	Employee_name User_id Email Role Business_unit Functional_unit Department Team Supervisor	Object Object Object Object Int64 Object Object Object Object

Table 4: Insider Threat Dataset r4.2 files' structure

5.2.4 Sample Data

We have sampled a portion of the data to start with, to experiment the system with smaller sizes of data frames in the process of building our model. When working with large datasets it is much easier to use smaller samples of the data, to troubleshoot functions and fix any errors in the code. After we have everything working with this sample data, we can use our model with the full dataset. A sample of the data, the first five (5) rows of each log files is shown in **Table 5** for device.csv, **Table 6** for file.csv, **Table 7** for email.csv, **Table 7**8for http.csv, **Table 9** for login.csv.

	id	date	User	pc	activity
0	{J1S3-L9UU75BQ-7790ATPL}	01/02/2010 07:21:06	MOH0273	PC-6699	Connect
1	{N7B5-Y7BB27SI-2946PUJK}	01/02/2010 07:37:41	MOH0273	PC-6699	Disconnect
2	{U1V9-Z7XT67KV-5649MYHI}	01/02/2010 07:59:11	HPH0075	PC-2417	Connect
3	{H0Z7-E6GB57XZ-1603MOXD}	01/02/2010 07:59:49	IIW0249	PC-0843	Connect
4	{L7P2-G4PX02RX-7999GYOY}	01/02/2010 08:04:26	IIW0249	PC-0843	Disconnect

Table 5: Sample of device.csv log

	id	date	user	pc	filename	content
0	{L9G8-J9QE34VM-2834VDPB}	01/02/2010 07:23:14	MOH0273	PC-6699	EYPC9Y08.doc	D0-CF-11-E0-A1-B1-1A-E1 during difficulty over...
1	{H0W6-L4FG38XG-9897XTEN}	01/02/2010 07:26:19	MOH0273	PC-6699	N3LTSU3O.pdf	25-50-44-46-2D carpenters 25 landed strait dis...
2	{M3Z0-O2KK89OX-5716MBIM}	01/02/2010 08:12:03	HPH0075	PC-2417	D3D3WC9W.doc	D0-CF-11-E0-A1-B1-1A-E1 union 24 declined impo...
3	{E1I4-S4QS61TG-3652YHKR}	01/02/2010 08:17:00	HPH0075	PC-2417	QCSW62YS.doc	D0-CF-11-E0-A1-B1-1A-E1 becoming period begin ...

	id	date	user	pc	filename	content
4	{D4R7-E7JL45UX-0067XALT}	01/02/2010 08:24:57	HSB0196	PC-8001	AU75JV6U.jpg	FF-D8

Table 6: Sample of file.csv log

	id	date	user	pc	to	cc	bcc	from	size	attachments	content
0	{R3I7-S4TX96FG-8219JWFF}	01/02/2010 07:11:45	LAP0338	PC-5758	Dean.Flynn.Hines@dtaa.com;Wade_Harrison@lockhe...	Nathaniel.Hunter.Heath@dtaa.com	NaN	Lynn.Adena.Pratt@dtaa.com	25830	0	middle f2 systems 4 july techniques powerfuld...
1	{R0R9-E4GL59IK-2907OSWJ}	01/02/2010 07:12:16	MOH0273	PC-6699	Odonnell-Gage@bellsouth.net	NaN	NaN	MOH68@optonline.net	29942	0	the breaking called allied reservations former...
2	{G2B2-A8XY58CP-2847ZJZL}	01/02/2010 07:13:00	LAP0338	PC-5758	Penelope_Colon@netzero.com	NaN	NaN	Lynn_A_Pratt@earthlink.net	28780	0	slowly this uncinus winter beneath addition ex...
3	{A3A9-F4TH89AA-8318GFGK}	01/02/2010 07:13:17	LAP0338	PC-5758	Judith_Hayden@comcast.net	NaN	NaN	Lynn_A_Pratt@earthlink.net	21907	0	400 other difficult land cirrocumulus powered ...

	id	date	user	pc	to	cc	bcc	from	size	attachments	content
4	{E8B7-C8FZ88UF-2946RUQQ}	01/02/2010 07:13:28	MOH0273	PC-6699	Bond-Raymond@verizon.net;Alea_Ferrell@msn.com;...	NaN	Odonnell-Gage@bellsouth.net	MOH68@optonline.net	17319	0	this kmh october holliswood number advised unu...

Table 7: Sample of email.csv log

	id	date	user	pc	url	content
0	{V1Y4-S2IR20QU-6154HFXJ}	01/02/2010 06:55:16	LRR0148	PC-4275	http://msn.com/The_Human_Centipede_First_Seque...	remain representatives consensus concert altho...
1	{Q5R1-T3EF87UE-2395RWZS}	01/02/2010 07:00:13	NGF0157	PC-6056	http://urbanspoon.com/Plunketts_Creek_Loyalsoc...	festival off northwards than congestion partne...
2	{X9O1-O0XW52VO-5806RPHG}	01/02/2010 07:03:46	NGF0157	PC-6056	http://aa.com/Rhodocene/rhodocenium/fhaavatqrf...	long away reorganized baldwin seth business 18...
3	{G5S8-U5OG04TE-5299CCTU}	01/02/2010 07:05:26	IRM0931	PC-7188	http://groupon.com/Leonhard_Euler/leonhard/tne...	among german schwein experimental becomes prev...
4	{L0R4-A9DH29VP-	01/02/2010 07:05:52	IRM0931	PC-7188	http://flickr.com/Inauguration_of_Barack_Obama...	kate criteria j 2008 highest 12 include books ...

id	date	user	pc	url	content
4553AUWM}					

Table 8: Sample of http.csv log

	id	date	user	pc	activity
0	{X1D9-S0ES98JV-5357PWMI}	01/02/2010 06:49:00	NGF0157	PC-6056	Logon
1	{G2B3-L6EJ61GT-2222RKSO}	01/02/2010 06:50:00	LRR0148	PC-4275	Logon
2	{U6Q3-U0WE70UA-3770UREL}	01/02/2010 06:53:04	LRR0148	PC-4124	Logon
3	{I0N5-R7NA26TG-6263KNGM}	01/02/2010 07:00:00	IRM0931	PC-7188	Logon
4	{D1S0-N6FH62BT-5398KANK}	01/02/2010 07:00:00	MOH0273	PC-6699	Logon

Table 9: Sample of logon.csv log

5.2.5 Initial Feature Selection

Since the dataset we employed to evaluate our system comes with specific scenarios, mentioned in section 5.2.2, we initially collected features based on these scenarios:

- For scenario #1, we need to detect an internal actor who logs on after hours, uses a removable device and uploads data to wikileaks.org. To detect the insider threat, we need specific features from device.csv, logon.csv, file.csv and http.csv.

- For scenario #2, we need to detect an internal actor who uses a thumb drive to steal data before leaving the company and join a competitor. To detect this insider threat, we need specific features from device.csv, file.csv and http.csv.

- For scenario #3, we need to detect a disgruntled system administrator, who downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. He collects his supervisor's credentials and uses them to send an alarming email to cause panic in the organization. To detect this insider threat, we need specific features from all csv files.

We have dropped features, which we believe are unnecessary in the detection of the above threats. **Table 10** presents the selected features from the csv files, along with the scenarios used for each feature.

File	Column	Datatype	Scenarios Used
Device.csv	Date	Object	1, 2, 3
	User	Object	1, 2, 3
	Pc	Object	1, 2, 3
	activity	Object	1, 2, 3
Email.csv	Date	Object	3
	User	Object	3
	Pc	Object	3
File.csv	Date	Object	1, 2, 3
	User	Object	1, 2, 3
	Pc	Object	1, 2, 3
http.csv	Date	Object	1, 2, 3
	User	Object	1, 2, 3
	Pc	Object	1, 2, 3
	url	Object	1, 2, 3
Logon.csv	Date	Object	1, 2, 3

	User	Object	1, 2, 3
	Pc	Object	1, 2, 3
	Activity	Object	1, 2, 3

Table 10: Selected Features based on dataset release's scenarios

Date values were collected as they were, but we had to split and encode the date feature into two features, day and time, since our system must detect outliers based on work after hours. 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 11**, along with their value space.

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, 2, 3, 4, 5, 6, 7

Table 11: Encoded Features

The values for the activity feature presented in **Table 11**, correspond to the user's activities, such as logon, logoff, connect disconnect, http, file and e-mail. An example dataset comprised of the beforementioned features is shown in **Table 12**.

Day	Time	User	PC	Activity
5	2	4512	4512	3

Table 12: Example Dataframe

We encoded categorical features using the One-Hot Encoding scheme at a later stage, since Machine Learning algorithms are more effective in prediction when working with datasets encoded using this scheme. In this scheme, we create a binary column for each category, based on the unique value of each feature. We transformed the "Activity" and generated seven (7) different features, shown in **Table 13**, with two possible values, 0 and 1. 0 indicates the absence of the specific activity and 1 indicates the presence of the same activity (sklearn.preprocessing.OneHotEncoder, 2020; What is One Hot Encoding? Why And When do you have to use it?, 2020). Finally, we joined all data into a pandas data frame and sorted it based on timestamp.

Feature	Possible Values
Day	0, 1, 2, 3, 4, 5, 6
Time	1, 2, 3, 4, …, 24
User	-
PC	-
Logon	0, 1
Logoff	0, 1
Connect	0, 1
Disconnect	0, 1
email	0, 1

File	0, 1
http	0, 1

Table 13: One-Hot Encoding

5.3 Feature Selection using Bio-Inspired algorithms

In EvoloPy-FS framework we defined the optimizers used by setting their flag in the main configuration to "True". In the main configuration, shown in **Figure 3**, we enabled PSO, GWO and BAT optimizers which correspond to Binary Particle Swarm Optimization, Binary Gray Wolf Optimizer and Binary Bat Algorithm, respectively. We also set the population size to 20 and iteration to 20 as well.

```
# Select optimizers
PSO= True
MVO= False
GWO = True
MFO= False
WOA= False
FFA=False
BAT=True



optimizer=[PSO, MVO, GWO, MFO, WOA,FFA,BAT]
datasets=["try01"]
#benchmarkfunc=[Fs1,Fs2,Fs3,Fs4,Fs5,Fs6,Fs7,Fs8,Fs9,Fs10]

# Select number of repetitions for each experiment.
# To obtain meaningful statistical results, usually 30 independent runs
# are executed for each algorithm.
NumOfRuns=1

# Select general parameters for all optimizers (population size, number of iterations)
PopulationSize = 20
Iterations= 20

#Export results ?
Export=True
```

Figure 3: EvoloPy-FS main configuration, "main.py".

5.4 Anomaly Detection (LOF)

As mentioned in section 4.4, out system utilizes the Local Outlier Factor (LOF), to detect the outlier or rare instances from the given data. In LOF we need to define the values of n_neighbors, contamination and n_jobs (Pedregosa et al., 2011):

- 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 data set. 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 to detect outliers. Python coding for Anomaly detection presented in Appendix A, section A.3.

5.5 Summary

In this chapter we presented our proposed system in detail, we described the data that was used and explained all relevant steps, from data collection, to data pre-processing, then to feature selection optimization and finally the malicious behavior detection.

Chapter 6
Findings and Analysis

In this chapter we present the 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. We compare our model's performance with other approaches.

6.1 Experimental Setup

All data processing tasks for this M.Sc. dissertation 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.

6.2 Testing the Model

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

We started the experiment by fitting the Machine learning model to the synthetic dataset described in subsection 5.2.1. We loaded the full dataset at the beginning of the experiments, as shown in **Figure 4**, but due to high memory usage we dropped these dataframes and created samples of the first 50000 rows from the dataset, as shown in **Figure 5**, to work with a smaller portion of it.

```
#load full dataset files into dataframes
logins = pd.read_csv('d:\\master\\r4.2\\logon.csv', usecols=['date', 'user', 'pc', 'activity'])

devices = pd.read_csv('d:\\master\\r4.2\\device.csv', usecols=['date', 'user', 'pc', 'activity'])

files = pd.read_csv('d:\\master\\r4.2\\file.csv', usecols=['date', 'user', 'pc'])

http = pd.read_csv('d:\\master\\r4.2\\http.csv', usecols=['date', 'user', 'pc'])

email = pd.read_csv('d:\\master\\r4.2\\email.csv', usecols=['date', 'user', 'pc'])
```

Figure 4: Load full dataset into pandas dataframes

```
#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 5: create samples from the dataset

Data collection and data pre-processing code execution takes place in order to prepare the dataframe with all selected features and fit the feature selection optimization to it. All relevant Python coding for Data collection and data pre-processing is presented in Appendix A, section A.1.

We fitted the feature selection optimization framework on the first thousand (1000) rows of the generated data with the results shown in **Table 14**. The three columns represent the results for each optimizer from the selected, PSO, GWO and BAT. Time taken is in seconds is about the same

for all three optimizers. Train and testing accuracy is computed based on the sliced part for each value, as shown in **Figure 6** and against the complete dataset. For each iteration run, we have a resulting objective fitness function which is the error rate and the number of selected features.

	PSO		**GWO**		**BAT**	
Time taken (seconds)	14.1658384799957		12.1430022716522		13.4572482109069	
Train Accuracy	0.993939393939393		1		0.809090909090909	
Test Acuracy	1		1		0.835294117647058	
Iter1	0.01315	7	0.0075	6	0.0075	6
Iter2	0.0075	6	0.0075	6	0.0075	6
Iter3	0.0075	8	0.0075	6	0.0075	6
Iter4	0.0075	6	0.0075	6	0.0075	6
Iter5	**0.00625**	5	0.0075	6	0.0075	6
Iter6	**0.00625**	4	0.0075	6	0.0075	6
Iter7	**0.00625**	7	0.0075	6	0.0075	6
Iter8	**0.00625**	6	0.0075	6	0.0075	5
Iter9	**0.00625**	4	0.0075	6	0.0075	3
Iter10	**0.00625**	6	0.0075	6	0.0075	5

Iter11	**0.00625**	4	0.0075	6	0.0075	3
Iter12	**0.00625**	6	0.0075	6	0.0075	4
Iter13	**0.00625**	4	0.0075	6	0.0075	3
Iter14	**0.00625**	4	0.0075	6	0.0075	2
Iter15	**0.00625**	6	0.0075	6	0.0075	1
Iter16	**0.00625**	5	0.0075	6	0.0075	1
Iter17	**0.00625**	6	0.0075	6	0.0075	1
Iter18	**0.00625**	7	0.0075	6	0.0075	1
Iter19	**0.00625**	7	0.0075	6	0.0075	1
Iter20	**0.00625**	6	0.0075	6	0.0075	1

Table 14: Feature Selection Optimization Results.

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

Figure 6: Training % and Testing %

The results of **Table 14** show that BPSO registered the minimum error rate, compared to the other two optimizers. 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, as mentioned in section 4.6. Precision is the ratio of True Positives, to the total number of positive results, Precision = TP / (TP + FP), with 1 being the best value and 0 the worst.

The first measurement is done by selecting all features (Figure 7) as generated after the data pre-processing phase and see the results without feature selection optimization. Following we run

more experiments based on the results of feature selection optimization. The results of the experiments are presented in Table 15.

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

Figure 7: All Features Selected

	Experiment	Features	Time_Taken	TP	FP	FN	DR	Precision
1	1.0	9.0	8.604782	70.0	918.0	0.0	1.000000	0.070850
2	2.0	5.0	6.609488	69.0	910.0	1.0	0.985714	0.070480
3	3.0	**4.0**	**5.928126**	62.0	774.0	8.0	0.885714	**0.074163**
4	4.0	7.0	7.686987	70.0	918.0	0.0	**1.000000**	0.070850
5	5.0	6.0	6.994491	69.0	919.0	1.0	0.985714	0.069838

Table 15: LOF results with feature optimization. N_neighbours = 20 and contamination = auto

The first results show 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 14, n_neighbours value was set to 20 and contamination set to auto. As mentioned in section 5.4, contamination represents the proportion of outliers in the data set.

We changed the contamination value to 0.1, to get similar results when contamination was set to auto (results are shown in **Table 16**). The optimal subset remains the same, but precision value is still very low.

	Experiment	Features	Time_Taken	TP	FP	FN	DR	Precision
1	1.0	9.0	11.854724	70.0	925.0	0.0	1.000000	0.070352
2	2.0	5.0	6.610281	69.0	923.0	1.0	0.985714	0.069556

	Experiment	Features	Time_Taken	TP	FP	FN	DR	Precision
3	3.0	**4.0**	**5.990587**	65.0	837.0	5.0	0.928571	**0.072062**
4	4.0	7.0	7.884163	70.0	925.0	0.0	1.000000	0.070352
5	5.0	6.0	8.453545	70.0	927.0	0.0	1.000000	0.070211

Table 16: LOF results with feature optimization. N_neighbours = 20 and contamination = 0.1

By changing the contamination value to 0.01 we got slightly different results (shown in **Table 17**), compared with the previous two cases and in this case the better precision value is produced by the experiment #2, with a subset dataset of five (5) features. This subset dataset was produced by both PSO and BAT algorithms.

	Experiment	Features	Time_Taken	TP	FP	FN	DR	Precision
1	1.0	9.0	9.417012	59.0	658.0	11.0	**0.842857**	0.082287
2	2.0	5.0	6.663487	56.0	573.0	14.0	0.800000	**0.089030**
3	3.0	4.0	**5.407866**	55.0	645.0	15.0	0.785714	0.078571
4	4.0	7.0	8.115871	58.0	654.0	12.0	0.828571	0.081461
5	5.0	6.0	7.711550	56.0	730.0	14.0	0.800000	0.071247

Table 17: LOF results with feature optimization. N_neighbours = 20 and contamination = 0.01

In all above cases, a portion of the original dataset was used in order to detect outliers.

6.3 Performance Results

The findings of section 6.2, show that after feature selection optimization, one of the resulted subsets is a near-optimal subset, since it performs better of the original once, when measured with precision. This near-optimal subset was generated after feature selection optimization by using BPSO and BBAT optimizers. and by fitting the LOF algorithm to it, a detection of 62 True Positive malicious insiders is a good result.

Five iterations of BPSO with minimum error rate and one of BBAT resulted with four (4) number of selected features. These results acknowledge that Swarm intelligence algorithms have high performance on feature selection optimization problems and can be used to enhance Machine learning models.

6.4 Testing more Bio-Inspired models

In section 4.3 we mentioned the selection of three (3) optimizers, BPSO, BGWO and BBAT, based on our initial decision, mentioned in section 3.16, of employing and testing three (3) Bio-Inspired algorithms for feature selection optimization. In section 3.16 we justify our selection for the selection of the specific algorithms, taking into our concern the tests run by Khurma et al. (2020) on various datasets and the reported performance of the specific algorithms, but also the popularity and exploration for each algorithm in literature.

Since EvoloPy-FS framework contains additional models we can use them as well and test their performance on or dataset.

We enabled MVO, MFO, WOA, and FFA models, which corresponds to Multi-Verse Optimizer, Moth-Flame Optimization, Whale Optimization Algorithm and Firefly Algorithm, respectively (**Figure 8** displays the change on the Framework's configuration).

```
# Select optimizers
PSO= False
MVO= True
GWO = False
MFO= True
WOA= True
FFA=True
BAT=False



optimizer=[PSO, MVO, GWO, MFO, WOA,FFA,BAT]
#datasets=["try01"]
datasets=["df_first_1000_rows"]
#datasets=["BreastCancer"]
#datasets=["ionosphere","BreastCancer","iris"]
#benchmarkfunc=[Fs1,Fs2,Fs3,Fs4,Fs5,Fs6,Fs7,Fs8,Fs9,Fs10]

# Select number of repetitions for each experiment.
# To obtain meaningful statistical results, usually 30 independent runs
# are executed for each algorithm.
NumOfRuns=1

# Select general parameters for all optimizers (population size, number of iterations)
PopulationSize = 20
Iterations= 20

#Export results ?
Export=True
```

Figure 8: EvoloPy-FS main configuration, "main.py" – test more models.

We fitted the feature selection optimization framework, configured with the additional optimizers, on the first thousand (1000) rows of the generated data with the results shown in **Table 18.** We run a number of tests, since EvoloPy-FS recommends multiple executions to obtain meaningful results. All four additional tested optimizers had similar results regarding the number of selected features, with a selection of five (5) features. Additionally, the number of four (4) features selected by all four (4) optimizers on a few iterations, two for MVO, one for MFO, one for WOA and three for FFA. These results show that the near-optimal subset can be generated by using these optimizers as well and proves the effectiveness of Bio-Inspired models on solving optimization problems.

	MVO	**MFO**	**WOA**	**FFA**
No. of features selected on most tests and iterations	5 features	5 features	5 features	5 features

No. of Iterations we had four selected features	2 Iterations	1 iteration	1 iteration	3 Iterations

Table 18: Feature Selection Optimization Results – Additional Models.

6.5 Performance Comparison with Other Approaches

We compared our approach with approaches that used Hidden Markov Models (HMM), Damerau–Levenshtein (DL) Distance, Cosine Distance, and Jaccard Distance techniques (Table 18 reports the performance of each approach based on TP/FP detection rate).

Rashid, Agrafiotis and Nurse (2016), used HMM and reported an 85% identification rate with a false positive rate of 20%. Lo et al. (2018) acknowledge that during the training phase of HMM the computational time can be quite slow as the number of features increases. Lo et al. (2018), used HMM and distance measurement techniques and reported a detection rate of 69% and 80& (aggregate score), respectively. The authors reported that HMM took more than 24 hours to process all data, in opposition with 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 high number of FP, it also reports better TP identification rate compared with other approaches as reported in **Table 19**.

Author	Approach	TP	FP
This M.Sc. dissertation	Exp. 3 Table 15	93%	90%
This M.Sc. dissertation	Exp. 3 Table 16	88.5%	83%
Rashid, Agrafiotis and Nurse (2016)	HMM	85%	20%

Lo et al. (2018)	DL	39%	
Lo et al. (2018)	Jaccard	36%	
Lo et al. (2018)	Cosine	47%	
Lo et al. (2018)	DL, Jaccard & Cosine aggregate score	80%	
Lo et al. (2018)	HMM	69%	

Table 19: Comparison with other approaches

6.6 Summary

In this chapter we presented our experiments' results along with our proposed model's performance and a comparison with other approaches. Our model reports a high number of FP, compared with other approaches, but also reports better TP identification rate than the compared approaches.

Chapter 7
Conclusion

In this M.Sc. dissertation 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 three 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 from the original one and can improve the performance of a machine learning model.

The employment of labeled data and addition of extra features, such as 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 on the detection of malicious insiders and reduce FP rate. These additional indicators must be transparent to all internal actors who must be clearly informed about all their log activity, according to

regulations of General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). The collection of employee's private data in an organization, might raise several issues and concerns and eventually reduce productivity.

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

7.1 Limitations

The process of detecting malicious insiders by detecting significant changes, or anomalies from 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 does not work for them.

7.2 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 firefly algorithm (FFA) and compare their performance with the Bio-Inspired models we experiment with in this M.Sc. dissertation.

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.

Bibliography

[01] Aljarah, I., Mafarja, M., Heidari, A.A., Faris, H., Zhang, Y. and Mirjalili, S. (2018). Asynchronous accelerating multi-leader salp chains for feature selection. Applied Soft Computing, 71, pp.964-979.

[02] Alpaydin, E. (2014). Introduction to Machine Learning. 3rd ed.

[03] Anaconda Software Distribution. Computer software. Vers. 3.0. Anaconda (2019). Available at <https://anaconda.com>

[04] Axelrad, E.T., Sticha, P.J., Brdiczka, O. and Shen, J., (2013). A Bayesian network model for predicting insider threats. In 2013 IEEE Security and Privacy Workshops (pp. 82-89). IEEE.

[05] Bai, Q. (2010). Analysis of particle swarm optimization algorithm. Computer and information science, 3(1), 180.

[06] Breunig, M.M., Kriegel, H.P., Ng, R.T. and Sander, J. (2000). LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 93-104).

[07] Brown, C.R., Watkins, A. and Greitzer, F.L., (2013). Predicting insider threat risks through linguistic analysis of electronic communication. In 2013 46th Hawaii International Conference on System Sciences (pp. 1849-1858). IEEE.

[08] Chakraborty, A. and Kar, A. (2017). Swarm Intelligence: A Review of Algorithms. Nature-Inspired Computing and Optimization, pp.475-494.

[09] Cui, Z. and Cai, X. (2013). Artificial Plant Optimization Algorithm. Swarm Intelligence and Bio-Inspired Computation, pp.351-365.

[10] Cui, W. and He, Y. (2018). Biological Flower Pollination Algorithm with Orthogonal Learning Strategy and Catfish Effect Mechanism for Global Optimization Problems. Mathematical Problems in Engineering, 2018, pp.1-16.

[11] Darwish, A. (2018). Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. Future Computing and Informatics Journal, 3(2), pp.231-246.

[12] Das S., Biswas A., Dasgupta S., Abraham A. (2009) Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications. In: Abraham A., Hassanien AE., Siarry P., Engelbrecht A. (eds) Foundations of Computational Intelligence Volume 3. Studies in Computational Intelligence, vol 203. Springer, Berlin, Heidelberg

[13] del Valle, Y., Venayagamoorthy, G., Mohagheghi, S., Hernandez, J. and Harley, R. (2008). Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. IEEE Transactions on Evolutionary Computation, 12(2), pp.171-195.

[14] Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) (Vol. 2, pp. 1470-1477). IEEE..

[15] Dictionary.cambridge.org. (2019). MITIGATE | meaning in the Cambridge English Dictionary. [online] Available at: https://dictionary.cambridge.org/dictionary/english/mitigate [Accessed 6 Sep. 2019].

[16] Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. Theoretical Computer Science, 344(2-3), pp.243-278.

[17] Eldardiry, H., Bart, E., Liu, J., Hanley, J., Price, B. and Brdiczka, O. (2013). Multi-Domain Information Fusion for Insider Threat Detection. 2013 IEEE Security and Privacy Workshops, pp. 45-51.

[18] Emary, E., Zawbaa, H., Ghany, K., Hassanien, A. and Parv, B. (2015). Firefly Optimization Algorithm for Feature Selection. Proceedings of the 7th Balkan Conference on Informatics Conference - BCI '15,.

[19] Emary, E., Zawbaa, H. and Hassanien, A. (2016a). Binary ant lion approaches for feature selection. Neurocomputing, 213, pp.54-65.

[20] Emary, E., Zawbaa, H. and Hassanien, A. (2016b). Binary grey wolf optimization approaches for feature selection. Neurocomputing, 172, pp.371-381.

[21] Muzaffar Eusuff, Kevin Lansey & Fayzul Pasha (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Engineering Optimization, 38:2, 129-154.

[22] Fang, C. and Wang, L. (2012). An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. Computers & Operations Research, 39(5), pp.890-901.

[23] Faris, H., Aljarah, I., Mirjalili, S., Castillo, P.A. and Guervós, J.J.M. (2016). EvoloPy: An Open-source Nature-inspired Optimization Framework in Python. In IJCCI (ECTA) (pp. 171-177).

[24] Faris, H., Mafarja, M.M., Heidari, A.A., Aljarah, I., Ala'M, A.Z., Mirjalili, S. and Fujita, H. (2018). An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowledge-Based Systems, 154, pp.43-67.

[25] Faris, H., Heidari, A.A., Ala'M, A.Z., Mafarja, M., Aljarah, I., Eshtay, M. and Mirjalili, S. (2020). Time-varying hierarchical chains of salps with random weight networks for feature selection. Expert Systems with Applications, 140, p.112898.

[26] Ferreira, P., C. Le, D. and Zincir-Heywood, N., (2019). Exploring Feature Normalization and Temporal Information for Machine Learning Based Insider Threat Detection. 2019 15th International Conference on Network and Service Management (CNSM)

[27] Gandomi, A., Yang, X. and Alavi, A., (2011a). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers, 29(1), pp.17-35.

[28] Gandomi, A., Yang, X. and Alavi, A. (2011b). Mixed variable structural optimization using Firefly Algorithm. Computers & Structures, 89(23-24), pp.2325-2336.

[29] Garro, B., Sossa, H. and Vazquez, R. (2009). Design of artificial neural networks using a modified Particle Swarm Optimization algorithm. 2009 International Joint Conference on Neural Networks,.

[30] Ghanem, W. and Jantan, A. (2014). Swarm intelligence and neural network for data classification. 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014),.

[31] Glasser, J. and Lindauer, B. (2013). Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. 2013 IEEE Security and Privacy Workshops (pp. 98-104). IEEE.

[32] Greitzer, F.L., Strozer, J., Cohen, S., Bergey, J., Cowley, J., Moore, A. and Mundie, D. (2014). Unintentional insider threat: contributing factors, observables, and mitigation strategies. In 2014 47th Hawaii International Conference on System Sciences (pp. 2025-2034). IEEE.

[33] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, pp.1157-1182.

[34] Hackernoon.com. 2020. What Is One Hot Encoding? Why And When Do You Have To Use It?. [online] Available at: <https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f> [Accessed 22 March 2020].

[35] Hassan, R., Cohanim, B., de Weck, O. and Venter, G. (2005). A Comparison of Particle Swarm Optimization and the Genetic Algorithm. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference,.

[36] Jupyter.org. (2020). Project Jupyter. [online] Available at: <https://jupyter.org/> [Accessed 3 May 2020].

[37] Kar, A. (2016). Bio inspired computing – A review of algorithms and scope of applications. Expert Systems with Applications, 59, pp.20-32.

[38] Karaboga, D. and Basturk, B., (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3), pp.459-471.

[39] Kennedy, J. and Eberhart, R., (1995). Particle swarm optimization (PSO). In Proc. IEEE International Conference on Neural Networks, Perth, Australia (pp. 1942-1948)

[40] Kennedy, J., (2006). Swarm intelligence. In Handbook of nature-inspired and innovative computing (pp. 187-219). Springer, Boston, MA.

[41] Khurma R.A., Aljarah I., Sharieh A., Mirjalili S. (2020) EvoloPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection. In: Mirjalili S., Faris H., Aljarah I. (eds) Evolutionary Machine Learning Techniques. Algorithms for Intelligent Systems. Springer, pp.131-173.

[42] Kilic, H., Yuzgec, U. and Karakuzu, C. (2018). A novel improved antlion optimizer algorithm and its comparative performance. Neural Computing and Applications, 32(8), pp.3803-3824.

[43] Kolias, C., Kambourakis, G. and Maragoudakis, M. (2011). Swarm intelligence in intrusion detection: A survey. Computers & Security, 30(8), pp.625-642.

[44] Koutsouvelis, V., Shiaeles, S., Keltoum, G., and Ghita, B. (2020). Detection of Insider Threats using Artificial Intelligence and Visualisation. In 2020 IEEE Conference on Network Softwarization (NetSoft). IEEE.

[45] Krishnanand, K.R., Nayak, S.K., Panigrahi, B.K. and Rout, P.K. (2009). Comparative study of five bio-inspired evolutionary optimization techniques. In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC) (pp. 1231-1236). IEEE.

[46] Le, D.C. and Zincir-Heywood, A.N., (2019). Machine learning based insider threat modelling and detection. In 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 1-6). IEEE.

[47] Li X., Clerc M. (2019) Swarm Intelligence. In: Gendreau M., Potvin JY. (eds) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol 272. Springer, Cham, pp.353-384.

[48] Liu, L., De Vel, O., Han, Q., Zhang, J. and Xiang, Y. (2018). Detecting and Preventing Cyber Insider Threats: A Survey. IEEE Communications Surveys & Tutorials, 20(2), pp.1397-1417.

[49] Lo, O., Buchanan, W., Griffiths, P. and Macfarlane, R. (2018). Distance Measurement Methods for Improved Insider Threat Detection. Security and Communication Networks, 2018, pp.1-18.

[50] Mafarja, M., Aljarah, I., Heidari, A.A., Faris, H., Fournier-Viger, P., Li, X. and Mirjalili, S. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowledge-Based Systems, 161, pp.185-204.

[51] McKinney, W. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).

[52] Meyer-Baese, A. and Schmid, V. (2014). Genetic Algorithms. Pattern Recognition and Signal Analysis in Medical Imaging, pp.135-149.

[53] Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, pp.46-61.

[54] Mirjalili, S. (2015). The Ant Lion Optimizer. Advances in Engineering Software, 83, pp.80-98.

[55] Mohemmed, A., Zhang, M. and Browne, W. (2010). Particle swarm optimisation for outlier detection. Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10,.

[56] Nakamura, R., Pereira, L., Costa, K., Rodrigues, D., Papa, J. and Yang, X. (2012). BBA: A Binary Bat Algorithm for Feature Selection. 2012 25th

[57] Nurse, J.R., Buckley, O., Legg, P.A., Goldsmith, M., Creese, S., Wright, G.R. and Whitty, M., (2014), May. Understanding insider threat: A framework for characterising attacks. In 2014 IEEE Security and Privacy Workshops (pp. 214-228). IEEE.

[58] Oliphant, T.E. (2006). A guide to NumPy (Vol. 1, p. 85). USA: Trelgol Publishing.

[59] Papadaki. M. and Shiaeles. S. (2018). “Insider Threat: The forgotten, yet formidable foe”, Chapter in Human Computer Interaction in Cyber-Security Handbook, CRC-Press, Taylor & Francis Group.

[60] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, pp.2825-2830.

[61] Pfleeger, C. and Pfleeger, S. (2002). Security in Computing, Third Edition. Prentice Hall.

[62] Python Software Foundation (2019). Python Language Reference, version 3.7.3. Available at https://python.org

[63] Rashid, T., Agrafiotis, I. and Nurse, J. (2016). A New Take on Detecting Insider Threats. Proceedings of the 2016 International Workshop on Managing Insider Security Threats - MIST '16,.

[64] Saka, M., Doğan, E. and Aydogdu, I. (2013). Analysis of Swarm Intelligence–Based Algorithms for Constrained Optimization. Swarm Intelligence and Bio-Inspired Computation, pp.25-48.

[65] Salem M.B., Hershkop S., Stolfo S.J. (2008) A Survey of Insider Attack Detection Research. In: Stolfo S.J., Bellovin S.M., Keromytis A.D., Hershkop S., Smith S.W., Sinclair S. (eds) Insider Attack and Cyber Security. Advances in Information Security, vol 39. Springer, Boston, MA

[66] Sarle, W.S. (1994). Neural networks and statistical models.

[67] Scikit-learn.org. 2020. Sklearn.Preprocessing.Onehotencoder. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html> [Accessed 20 March 2020].

[68] Schultz, E. (2002). A framework for understanding and predicting insider attacks. Computers & Security, 21(6), pp.526-531.

[69] Sivanandam, S.N. and Deepa, S.N. (2008). Genetic algorithm optimization problems. In Introduction to genetic algorithms (pp. 165-209). Springer, Berlin, Heidelberg.

[70] Snyman, J. (2000). The LFOPC leap-frog algorithm for constrained optimization. Computers & Mathematics with Applications, 40(8-9), pp.1085-1096.

[71] Srinoy, S. (2007). Intrusion Detection Model Based On Particle Swarm Optimization and Support Vector Machine. 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications, pp. 186-192.

[72] Sun, N., Zhang, J., Rimba, P., Gao, S., Zhang, L. and Xiang, Y. (2019). Data-Driven Cybersecurity Incident Prediction: A Survey. IEEE Communications Surveys & Tutorials, 21(2), pp.1744-1772.

[73] Theis, Michael., Trzeciak, Randall., Costa, Daniel., Moore, Andrew., Miller, Sarah., Cassidy, Tracy., & Claycomb, William. (2019). Common Sense Guide to Mitigating Insider Threats, Sixth Edition (CMU/SEI-2018-TR-010). Retrieved February 09, 2020, from the Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=540644

[74] The pandas development team (2020). pandas-dev/pandas: Pandas ver. latest. Available at <https://doi.org/10.5281/zenodo.3509134>

[75] Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N. and Robinson, S. (2017). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In Workshops at the Thirty-First AAAI Conference on Artificial Intelligence

[76] Tzu, S. (2007). *The Art of War*. Filiquarian Pub.

[77] Verizon Enterprise, (2017). Verizon Data breach investigations repor 10th edition. [online] Available: https://www.verizonenterprise.com/verizon-insights-lab/dbir/ [Accessed 30 August. 2019]

[78] Verizon Enterpise, (2019). Verizon Data Breach Investigations Report 12th edition. [online] Available: https://www.verizonenterprise.com/verizon-insights-lab/dbir/ [Accessed 30 August. 2019]

[79] Wang, G. (2018). A Comparative Study of Cuckoo Algorithm and Ant Colony Algorithm in Optimal Path Problems. MATEC Web of Conferences, 232, p.03003.

[80] Wang, J. S., & Li, S. X. (2019). An improved grey wolf optimizer based on differential evolution and elimination mechanism. Scientific reports, 9(1), 1-21.

[81] S. Widup, M. Spitler, D. Hylender, G. Bassett, (2018). Verizon Data Breach Investigations Report 11th edition. [online] Available: https://www.verizonenterprise.com/verizon-insights-lab/dbir/ [Accessed 9 Feb. 2019]

[82] Xiao, L., Shao, Z. and Liu, G. (2006). K-means Algorithm Based on Particle Swarm Optimization Algorithm for Anomaly Intrusion Detection. 2006 6th World Congress on Intelligent Control and Automation, pp. 5854-5858.

[83] Yan, G. (2011). An Effective Refinement Artificial Bee Colony Optimization Algorithm Based On Chaotic Search and Application for PID Control Tuning.

[84] Yang X.S. (2010a) A New Metaheuristic Bat-Inspired Algorithm. In: González J.R., Pelta D.A., Cruz C., Terrazas G., Krasnogor N. (eds) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, vol 284. Springer, Berlin, Heidelberg

[85] Yang, X.S. (2010b). Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv:1003.1409.

[86] Yang XS. (2012). Flower Pollination Algorithm for Global Optimization. In: Durand-Lose J., Jonoska N. (eds) Unconventional Computation and Natural Computation. UCNC 2012. Lecture Notes in Computer Science, vol 7445. Springer, Berlin, Heidelberg

[87] Yang, X. and He, X. (2013a). Firefly algorithm: recent advances and applications. International Journal of Swarm Intelligence, 1(1), p.36.

[88] Yang, X. S., and He, X. (2013b). Bat algorithm: literature review and applications. International Journal of Bio-inspired computation, 5(3), 141-149.

[89] Yang, X., Karamanoglu, M. and He, X. (2013). Flower pollination algorithm: A novel approach for multiobjective optimization. Engineering Optimization, 46(9), pp.1222-1237.

[90] Yuan F., Cao Y., Shang Y., Liu Y., Tan J., Fang B. (2018) Insider Threat Detection with Deep Neural Network. In: Shi Y. et al. (eds) Computational Science – ICCS 2018. ICCS 2018. Lecture Notes in Computer Science, vol 10860, pp.43-54. Springer, Cham.

[91] Zhan, Z., Zhang, J., Li, Y. and Chung, H. (2009). Adaptive Particle Swarm Optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(6), pp.1362-1381.

Appendix A
Python Code

A.1 Data Collection & Data Pre-Processing

```
#import pandas data analysis library
import pandas as pd

#import numpy library to use for large arrays and mathematical functions
import numpy as np

#import time library for time access and conversions
import time as t

#The datetime.datetime class is the most commonly-used class from the datetime module
from datetime import datetime, date, time

#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)

#load sample files into dataframes
#load logins.csv into a pandas DataFrame
logins = pd.read_csv('d:\\master\\r4.2\\sample-logon.csv', usecols=['date', 'user', 'pc', 'activity'])

#load devices.csv into a pandas DataFrame
devices = pd.read_csv('d:\\master\\r4.2\\sample-device.csv', usecols=['date', 'user', 'pc',
'activity'])

#load files.csv into a pandas DataFrame
files = pd.read_csv('d:\\master\\r4.2\\sample-file.csv', usecols=['date', 'user', 'pc'])

#load http.csv into a pandas DataFrame
http = pd.read_csv('d:\\master\\r4.2\\sample-http.csv', usecols=['date', 'user', 'pc'])

#load email.csv into a pandas DataFrame
email = pd.read_csv('d:\\master\\r4.2\\sample-email.csv', usecols=['date', 'user', 'pc'])

#create activity columns for the files, http and email dataframes
files['activity'] = 6 #create activity column for files
http['activity'] = 7 #create activity column for http
email['activity'] = 5 #create activity column for http
```

```
#Log Aggregate
#Merge dataframes into one large dataframe
frames = [devices, files, logins, http, email]
df = pd.concat(frames)

#del initial data frames to save resources
del files, logins, devices, http, email

#convert all acticities in the dataset to integers
#Logon - 1
#Logoff - 2
#Connect - 3
#disconnect - 4
#email 5
#file - 6
#http 7

activities = ['Logon', 'Logoff', 'Connect', 'Disconnect']
i = 1
for activity in activities:
  df['activity'].replace(activity,i, inplace=True)
  i = i + 1
  del activity

del i, activities
############################################################

#Perform One Hot Encoding
df['Logon'] = 0
df['Logoff'] = 0
df['Connect'] = 0
df['Disconnect'] = 0
df['email'] = 0
df['file'] = 0
```

```
df['http'] = 0

#Perform One Hot Encoding
df.loc[df.activity == 1,"Logon"] = 1
df.loc[df.activity == 2,"Logoff"] = 1
df.loc[df.activity == 3,"Connect"] = 1
df.loc[df.activity == 4,"Disconnect"] = 1
df.loc[df.activity == 5,"email"] = 1
df.loc[df.activity == 6,"file"] = 1
df.loc[df.activity == 7,"http"] = 1

#convert date column to day and time https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html
df['date'] = pd.to_datetime(df['date'],infer_datetime_format=True)

#Split date column in day and time | #PARSE the date from strings
days , times = zip(*[(d.date().weekday(), float((d.time().minute/60)+d.time().hour)) for d in df['date']])
df = df.assign(day = days , time = times)
del days, times #save resources

#load the usernames of the malicious insiders according to the scenarios
insiders = ["AAM0658", "AJR0932", "BDV0168", "BIH0745", "BLS0678","BTL0226", "CAH0936", "DCH0843", "EHB0824", "EHD0584", "FMG0527","FTM0406", "GHL0460", "HJB0742", "JMB0308", "JRG0207", "KLH0596","KPC0073", "LJR0523", "LQC0479", "MAR0955", "MAS0025", "MCF0600","MYD0978", "PPF0435", "RAB0589", "RGG0064", "RKD0604", "TAP0551","WDD0366", "AAF0535", "ABC0174", "AKR0057", "CCL0068", "CEJ0109","CQW0652", "DIB0285", "DRR0162", "EDB0714", "EGD0132", "FSC0601","HBO0413", "HXL0968", "IJM0776", "IKR0401", "IUB0565", "JJM0203","KRL0501", "LCC0819", "MDH0580", "MOS0047", "NWT0098", "PNL0301","PSF0133", "RAR0725", "RHL0992", "RMW0542", "TNM0961", "VSS0154","XHW0498", "BBS0039", "BSS0369", "CCA0046", "CSC0217", "GTD0219","JGT0221", "JLM0364", "JTM0223", "MPM0220", "MSO0222"]

#create the column malicious to indicate if a user is malicious or not
```

```
df['malicious'] = 0
#if the user is malicious set feature to 1
df.loc[(df['user'].isin(insiders)),"malicious''] = 1

#sort dataset by date
#(https://www.geeksforgeeks.org/python-pandas-dataframe-sort_values-set-1/)
df.sort_values('date', axis=0, inplace=True)
```

A.2 Feature Selection Optimization

```
#create samples of the first 1000 rows of the dataframe for FS optimization
header = ["day", "time", "Logon", "Logoff", "Connect", "Disconnect", "email", "file", "http"]
df_first_1000_rows = df.head(1000)

#export pandas dataframes to csv to load into the FS optimization framework
df_first_1000_rows.to_csv('df_first_1000_rows.csv', columns = header, header = False, index=False)
```

Latest EvoloPy-FS Framework code used for feature selection optimization can be obtained from https://github.com/aljarrahcs/EvoloPy-FS

A.3 Anomaly Detection

```
#import LOF from scikit-learn
from sklearn.neighbors import LocalOutlierFactor

#import LOF from scikit-learn
from sklearn.neighbors import LocalOutlierFactor

#experiment 1 - select all features
selected_features = ['day', 'time', 'Logon', 'Logoff', 'Connect', 'Disconnect', "email", "file", "http"]

#experiment 2
selected_features = ['day', 'time', 'Logon', 'Connect', 'file']

#experiment 3
selected_features = ['day', 'time', 'Logon', 'Connect']

#experiment 4
selected_features = ['day', 'time', 'Logon', 'Logoff', 'Connect', 'Disconnect', "email" ]

#experiment 5
selected_features = ['day', 'time', 'Logon', 'Logoff', 'Connect', 'Disconnect']

#start timer
start_lof = t.time()

clf = LocalOutlierFactor(n_neighbors=10, contamination="auto", n_jobs=-1)
df['outlier'] = clf.fit_predict(df[selected_features])

#calculate execution time
end_lof = (t.time()-start_lof)

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

```
#selected features are based on the results of feature selection optimization

#start timer
start_lof = t.time()

#Predict outliers based on selected features
clf = LocalOutlierFactor(n_neighbors=20, contamination="auto", n_jobs=-1)
df['outlier'] = clf.fit_predict(df[selected_features])

#calculate execution time
end_lof = (t.time()-start_lof)


#LOF results
lof_results = df[df['outlier'] == -1].groupby(['user','malicious']).size().reset_index(name='frequency').sort_values('frequency', ascending=False)

#True Positive
predicted_insiders = lof_results[lof_results['malicious'] == 1].shape[0]

#False Positive
false_insiders = lof_results[lof_results['malicious'] == 0].shape[0]
```

A.4 Performance Metrics

```
#Table with performance metrics

#initialize columns

columns = ['Experiment','Features', 'Time_Taken', 'TP', 'FP', 'FN', 'DR', 'Precision']

performance_metrics = pd.DataFrame(columns=columns)



#append experiment results to table

performance_metrics.loc[experiment] = [experiment, len(selected_features), end_lof, predicted_insiders, false_insiders, (70-predicted_insiders), (predicted_insiders / 70), (predicted_insiders / (predicted_insiders + false_insiders))]



#show performance metrics table

performance_metrics
```