Machine Learning for Monitoring Attacks in the Cloud

Nils van Noort
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
n.vannoort@student.utwente.nl

ABSTRACT

'Cloud' is a computing infrastructure that is used for storage and processing. Many organizations have switched to using cloud, of which 28% experienced cloud security incidents in 2019 alone. One promising method for monitoring cloud attacks is using Machine Learning algorithms (ML). The problem is that it is not clear which ML approaches are suitable and most efficient for monitoring which cloud attack. While some related works have focused on using ML for monitoring cloud attacks, these studies covered only a handful of attacks. In contrast, in this research, we will cover ground towards broad use of ML for monitoring many different cloud attacks. Our methodology is threefold: (i) identifying cloud attacks, then (ii) recognizing problems that ML solves, and finally (iii) conducting a performance evaluation on ML using real data. Our analysis shows that ML-based monitoring can attain a 100% classification accuracy. Our contribution entails a foundation for future research.

Keywords

Machine learning, Cloud attacks, MITRE ATT&CK, Security, Monitoring

1. INTRODUCTION

'Cloud' has been used as a metaphor for the Internet. Overall, cloud refers to data centers used for remote storage and/or computing. Cloud is becoming increasingly popular with organizations as it promises more agility, efficiency and scalability compared to on-premise infrastructure. Gartner [1] projects the cloud services industry to grow exponentially through 2022. As organizations depend more and more on cloud, the importance of a secure cloud increases. This is evident from a recent $(ISC)^2$ Report[2]: 28% of organizations experienced a cloud attack of some sort in the year 2019 alone. Victims of cloud attacks include Facebook in 2019, where 146 GB of sensitive user data was leaked from an S3 bucket[3], part of an Amazon cloud service. Attacks in the cloud come in many flavours. For example, a well known cloud attack entails stealing cookies to become authenticated as another user, i.e. the Web Session Cookie attack[4]. Machine Learning algorithms (ML) are a promising alternative to traditional algorithms for monitoring these attacks.

ML is an old concept as it originates from the 1950s[5]. As computational power is increasing so rapidly, ML is receiving an increasing amount of attention from the community. Some[6] might say it is merely a 'buzzword' as many companies claim to be investing heavily into ML[7], but it is not openly explained what algorithms are used. The scientific community has been looking at many ways to use the benefits of ML. In fact, ML has been proven to be useful in many areas[8][9], including cloud security[10][11].

The problem we address in our research is that, from a literature survey, it seems existing works are merely scratching the surface when it comes to ML capabilities in a cloud environment. Previous works are often focused on using ML for a small subset of attacks. This information gap results in a situation where it is unclear what ML to use for monitoring cloud attacks.

The goal of our research is to bridge this gap towards improved monitoring of cloud attacks using ML. We provide a foundation towards a broad state-of-the-art overview, which entails what ML is suitable and efficient for monitoring which cloud attack. To pursue our goal, the following research questions (RQ) have been defined as the basis of our research:

RQ1 What problem types are associated with attacks against cloud resources?

RQ2 Which ML could be used for monitoring attacks in the cloud?

RQ3 How well does ML perform when monitoring cloud attacks?

Each RQ has its own dedicated section in this research paper as part of our methodology. In § 2 we discuss the attacks against cloud resources in order to answer RQ1. After that, § 3 focuses on answering RQ2. Here, we survey ML and connect their usage to the characteristics of the found cloud attacks. To answer RQ3, § 4 contains a performance evaluation of various ML algorithms. This is done in a real environment to a closed set of attacks using real data. Finally § 5 contains a conclusion where we summarize our findings.

Our performance evaluation has shown very positive results. Of the various ML algorithms we tested, we found that some of them performed extraordinarily well: The most promising result is that we managed a 100% classification accuracy when monitoring one of the cloud attacks we surveyed in this research. With our thorough methodology in place, we aim to incentivize further research on using ML for monitoring cloud attacks.

33rd Twente Student Conference on IT July 3rd, 2020, Enschede, The Netherlands.

2. SURVEYING CLOUD ATTACKS

The intent of this section is to detail the cloud attacks that we will be evaluating for ML monitoring. The attacks have been surveyed and analyzed so we can answer questions such as 'what cloud service is vulnerable to this attack?' and 'what problem type is associated with this attack?'. Ultimately, we want to answer and present which ML algorithms are suitable for monitoring each attack.

2.1 Methodology

To answer RQ1 (*What problem types are associated with attacks against cloud resources?*) we have performed a literature survey. A traditional literature study based on key-words and sources of data would have led to many unrelated attacks, where classification could have become problematic in the time-frame of this research. Instead, our methodology relies on using an open, globally-accessible framework that is well known in the security industry, namely MITRE ATT&CK[12]. This framework is proposed by a collaborative foundation for the development of attack information and is used in many sectors. The MITRE ATT&CK enterprise cloud attack section contains information on 10 cloud attack tactics and 36 separate cloud attack techniques[13].

We have summarized the 36 MITRE ATT&CK cloud attacks in Table 1. An arbitrary Technique Identifier (TI) is assigned to the attack techniques to facilitate referencing. For each technique is shown which tactic(s) it is part of and which cloud services are vulnerable to this attack. The associated problem type is given in the last column.

The attacks (TI-5) 'Valid Accounts' and (TI-15) 'Brute Force' are highlighted in Table 1. Only these two attacks will be used for the performance evaluation in § 4, given the limited time-frame of this research. The main reason we highlight these attacks specifically is that they represent 5 out of the 10 tactics of cloud attacks, which will make our findings more relevant. The next subsection will expand upon what all 10 tactics entail.

2.2 Cloud Attack Tactics

As shown in Table 1, there are 10 different tactics for cloud attacks in the MITRE ATT&CK knowledge base on Enterprise Cloud[13]. A tactic can be seen as an attack category, consisting of one or more attack techniques. For each tactic we will describe its purpose with an example of an attack technique from the Cloud section.

2.2.1 *Initial Access*

The 'Initial Access' tactic is a category of attack that is about getting initial entry to a network. This tactic grants the attacker an opportunity to gain access to a network as a first step. Using other tactics afterwards they might be able to have access for a prolonged amount of time.

An example of an attack technique within this tactic entails stealing the credentials of a user or service in order to gain initial access, i.e. 'Valid Accounts'. The network access could allow the attacker to pivot across accounts and ultimately obtain admin privileges.

2.2.2 *Persistence*

The 'Persistence' tactic is an attack category that entails techniques that allow attackers to maintain access once they are in. These techniques serve to keep access even when systems are restarted, credentials are changed or in case of other obstacles.

A good example of persistence is the 'Redundant Access' technique. By using multiple remote access tools you maintain access even when one of the entryways is detected and/or mitigated.

2.2.3 *Privilege Escalation*

This tactic has the primary goal of reaching higher level privileges on a system or network. The tactic consists of only one technique for Cloud, the aforementioned 'Valid Accounts' technique (§ 2.2.1).

In the 'Valid Accounts' technique, the attacker has already entered the system or network using a compromised account. Thus, they are able to explore the system/network for vulnerabilities, e.g. misconfigurations or user accounts with root permission.

2.2.4 *Defence Evasion*

'Defence Evasion' is a tactic of which the focus is to avoid being detected while you have broken into the system or network. There are multiple ways to achieve this, e.g. disabling or uninstalling security software, or obfuscating malicious behaviour to be seen as part of trusted processes.

Examples of techniques include the earlier mentioned 'Redundant Access' (§ 2.2.2) and 'Valid Accounts' (§ 2.2.1). As you can see, the techniques are not unique to a tactic and can instead go across multiple. In particular, the 'Redundant Access' technique has its focus on redundancy; if one entryway is mitigated, there are still others. Here, not being detected obviously plays an important role as you need not be detected while finding other entryways.

2.2.5 *Credential Access*

The 'Credential Access' tactic entails ways to steal account names and passwords. Techniques used include keylogging or credential dumping. If a database is leaked containing hashed passwords, the associated accounts are at risk as regular 'Brute Force' mitigation is not an option.

'Brute force' is a well-known technique that most are familiar with; you try a username and password in hopes that it is correct. Most services have a limit on login tries. However, an attacker could instead use 'password spraying'. Here, a small set of common passwords is attempted on many different accounts. This is a way to avoid the account block after many failed attempts.

2.2.6 *Discovery*

The tactic of 'Discovery' is relevant when the attacker wants to explore the unfamiliar environment. The attacker looks to gain knowledge on the details of the system or network they are in. Once they know what parts they have access to they can more adequately decide how to act next.

An example of a technique is 'Cloud Service Discovery'. After an attacker gains access to a system, they go through the cloud services on the system to discover information about them. The methods used depend on the service type: platform-as-a-service (PaaS), infrastructure-as-a-service (IaaS), or software-as-a-service (SaaS).

2.2.7 *Lateral Movement*

The 'Lateral Movement' tactic involves the attacker trying to move through the environment. It consists of techniques the attacker can use to remotely access and control different parts of the network. They might use this tactic when their objective requires them to travel between multiple systems that have to be breached.

An example of how an attacker might gain access to another system is by using a stolen web session cookie to become authenticated as a valid user, i.e. the 'Web Session Cookie' technique. The attacker imports the cookie

into a custom browser and is authenticated for as long as the cookie is active.

2.2.8 *Collection*

This tactic consists of techniques related to the collection of data. The gathering of data is done so they can be exfiltrated (stolen) afterwards.

One of those techniques entails staging the collected data in a single directory. This location can be filled with multiple files or combined into one using methods like encryption or compression.

2.2.9 *Exfiltration*

'Exfiltration' entails the theft of data from a network without being detected. Common ways include packaging the data using compression so the exfiltration goes unnoticed.

One technique for exfiltrating data is by transferring the data to one or multiple other cloud accounts. By splitting up the download over multiple smaller transfers you avoid the monitoring of large file transfers to outside the network.

2.2.10 *Impact*

This tactic involves the manipulation, interruption and/or destruction of data and parts of a system. An attacker could manipulate a system in order to use machine resources, tamper with data or even destroy data.

A technique relevant to cloud is 'Resource Hijacking'. Here an attacker might leverage the computing power of a cloud service for resource intensive tasks, e.g. cryptocurrency mining. They might also target user endpoint systems. These compromises may impact system or service availability depending on the severity of the attack.

Table 1. Cloud attacks from MITRE ATT&CK

TI	Technique	Tactic	AWS & GCP & Azure	Azure AD	Office 365	SaaS	Problem Type
1	Drive-by Compromise	IA	✗	✗	✗	✓	Class.
2	Exploit Public-Facing Application	IA	✓	✗	✗	✗	Class., Cluster
3	Spearphishing Link	IA	✗	✗	✓	✓	Class., Cluster
4	Trusted Relationship	IA	✓	✗	✗	✓	Class.
5	**Valid Accounts**	IA, P, PE, DE	✓	✗	✗	✓	Class., Regr.
6	Account Manipulation	P, CA	✓	✓	✓	✗	Class., Regr.
7	Create Account	P	✓	✓	✓	✗	Class.
8	Implant Container Image	P	✓	✗	✗	✗	Class.
9	Office Application Startup	P	✗	✗	✓	✗	Class.
10	Redundant Access	P, DE	✓	✓	✓	✓	Class.
11	Application Access Token	DE, LM	✗	✗	✓	✓	Class.
12	Revert Cloud Instance	DE	✓	✗	✗	✗	Class.
13	Unused/Unsupported Cloud Regions	DE	✓	✗	✗	✗	Class.
14	Web Session Cookie	DE, LM	✗	✗	✓	✓	Class.
15	**Brute Force**	CA	✗	✓	✓	✓	Class.
16	Cloud Instance Metadata API	CA	✓	✗	✗	✗	Class.
17	Credentials in Files	CA	✓	✗	✗	✗	Class.
18	Steal Application Access Token	CA	✗	✓	✓	✓	Class.
19	Steal Web Session Cookie	CA	✗	✗	✓	✓	Class.
20	Account Discovery	D	✗	✓	✓	✗	Class.
21	Cloud Service Dashboard	D	✓	✓	✓	✗	Class.
22	Cloud Service Discovery	D	✓	✓	✓	✓	Class., Regr.
23	Network Service Scanning	D	✓	✗	✗	✗	Class.
24	Network Share Discovery	D	✓	✗	✗	✗	Class.
25	Permission Groups Discovery	D	✗	✓	✓	✗	Class.
26	Remote System Discovery	D	✓	✗	✗	✗	Class.
27	System Information Discovery	D	✓	✗	✗	✗	Class.
28	System Network Conn. Discovery	D	✓	✗	✗	✗	Class.
29	Internal Spearphishing	LM	✗	✗	✓	✓	Class., Cluster
30	Data from Cloud Storage Object	C	✓	✗	✗	✗	Class., Regr.
31	Data from Information Repositories	C	✓	✗	✗	✓	Class., Regr.
32	Data from Local System	C	✓	✗	✗	✗	Class.
33	Data Staged	C	✓	✗	✗	✗	Class.
34	Email Collection	C	✗	✗	✓	✗	Class.
35	Transfer Data to Cloud Account	E	✓	✗	✗	✗	Class.
36	Resource Hijacking	I	✓	✗	✗	✗	Class., Regr.

IA: Initial Access, P: Persistence, PE: Privilege Escalation, DE: Defence Evasion, CA: Credential Access, D: Discovery, LM: Lateral Movement, C: Collection, E: Exfiltration, I: Impact

2.3 Findings

We have surveyed all attacks from MITRE ATT&CK Cloud Enterprise. For each attack we have summarized characteristics in Table 1, including the problem type associated with this attack. This problem type is very relevant to the next section of this paper, as this is one of the parameters we used to evaluate which ML algorithms would be most suitable for monitoring that specific attack technique.

What is striking about this overview is that it shows the type of problem to solve for all techniques: We have found that all cloud attack techniques require classification of some sort. In hindsight this does make sense as monitoring for attacks often involves identifying what actions are part of an attack versus normal user actions. These groups ('normal'/'attack') are the labels for the binary classification problem. We will expand upon what classification algorithms could be used for monitoring these attacks in the next section.

3. SURVEYING MACHINE LEARNING

This section covers the ML approaches that are relevant to monitoring attacks in the cloud. We focus on the type of ML to be able to identify appropriate algorithms. We have surveyed related works on ML that could be useful for this identification.

3.1 Methodology

To solve RQ2 (*Which ML could be used for monitoring attacks in the cloud?*) we must look at which ML to use for monitoring the various attacks from RQ1. According to Wolpert and Macready [14]'s famous 'No Free Lunch Theorem', 'no single algorithm works for all problems.' As such we had to evaluate what ML algorithms are suitable for each individual attack technique.

At the bottom of this page is Table 2 which shows which ML algorithms were used in works that are closely related to the kind of problem associated with monitoring cloud attacks. With reference to both related works and the problem type of an attack, we can possibly deduct which ML algorithms are suitable to a certain attack. Some of the related works include cloud attacks, but some do not; the goal here is to use the knowledge from a different but relevant perspective to apply it in this research.

The related works each have a specific focus; they address a certain attack or other (classification) problem. The way that our research differs from past works is that we aim to provide a large overview of what ML algorithm to use for monitoring which cloud attack. The amount of algorithms used in these works differs, but most fail to cover all the common classification algorithms. This leaves possible performance on the table as an untested algorithm might have outperformed the others.

There are several paradigms in ML, e.g. Supervised, Unsupervised, and Reinforcement Learning. For the use case of monitoring cloud attacks, it seems logical to use a supervised ML approach. From what we found, monitoring cloud attacks very often involves differentiating between 'attack' and 'normal' actions/processes. This kind of problem lends itself well to a supervised binary classification approach. We have surveyed a number of well-established ML classification algorithms from related works, which we detail below.

3.1.1 *Support Vector Machine (SVM)*

Support Vector Machine (SVM) is a ML algorithm that involves drawing a line (hyperplane) between two data classes which effectively divides the two. According to Kotsiantis [22], 'Maximizing the margin and thereby creating the largest possible distance between the separating hyperplane and the instances on either side of it has been proven to reduce an upper bound on the expected generalisation error.' When predicting the class of a data point the algorithm evaluates which side of the hyperplane the point is on.

Some data sets are not separable by a linear hyperplane. Kotsiantis goes on to state that for most real world problems no hyperplane exists that successfully separates the data points of different classes.[22] In this case, using higher dimension SVM might be better for solving the inseparability problem. For this research, we will evaluate the performance of linear SVM when monitoring attacks.

3.1.2 *Naive Bayes*

The Naive Bayes algorithm assumes that all features are independent of one another. 'For this reason naive Bayes classifiers are usually less accurate that other more sophisticated learning algorithms', according to Kotsiantis [22]. However they also state that in some cases it can be superior to other learning schemes.

A calculation is made using the Bayes Theorem[23] and a product operation. The probabilities of instances belonging to a class are compared, where the highest probability decides the predicted class. Naive Bayes has a major advantage over other classifiers, namely the short computational time necessary for training[22].

3.1.3 *k-Nearest Neighbours (K-NN)*

The K-NN algorithm is quite simple in how it works. It relies on the fact that 'the instances within a dataset will generally exist in close proximity to other instances that have similar properties'[22]. When evaluating a new data point it evaluates an area around it with a radius of 'k', which is a parameter that can be tweaked to get the best performance. Within this area it counts the amount of occurrences of each class, and assigns the most occurring class to the original data point.

Table 2. ML Algorithms utilized in Related Works

ref	Logistic Regr.	k-Nearest Neighbours	Decision Trees	SVM	Naive Bayes	LDA	k-Means Unsuperv.	Problem addressed
[15]	✓	✗	✓	✓	✓	✗	✓	Malicious URL detection
[16]	✗	✓	✓	✓	✓	✗	✗	Class. of text-documents
[17]	✗	✗	✗	✗	✓	✗	✗	Classifying internet traffic
[18]	✗	✗	✗	✗	✗	✗	✓	Deep packet inspection
[19]	✓	✗	✗	✓	✗	✗	✗	Detecting bot accounts
[20]	✓	✓	✓	✓	✓	✗	✗	XXS attack detection
[21]	✗	✗	✗	✓	✗	✗	✗	SQL injection attacks
	Class./Regr.	Class.	Class.	Class.	Class.	Class.	Cluster	

Unlike most other classifiers, K-NN doesn't have a set 'training phase' because it stores the entire training set to make predictions about subsequent data points[24]. As such, a major drawback is the storage requirement when using this algorithm[22]. Another drawback of K-NN is low performance when the features are categorical, as it is hard to define a 'distance' between categories[24].

3.1.4 *Decision Trees*

The Decision Trees classifier relies on a rule-based flow of decisions, where each decision narrows down what class the instance is likely to be part of. A decision is made according to the value of a feature, and 'the feature that best divides the training data would be the root node of the tree'[22]. However, Murthy [25] notes that a majority of studies have found that there is no single best method for determining the root.

There is a possible danger of overfitting when using the Decision Tree algorithm. To this end, Kotsiantis [22] states that 'if two trees employ the same kind of tests and have the same prediction accuracy, the one with fewer leaves is usually preferred'. On this topic, Breslow and Aha [26] have surveyed various methods for tree simplification.

3.1.5 *Logistic Regression*

Logistic Regression is a type of regression that behaves like a classifying algorithm. 'A very convenient and useful side effect of a logistic regression solution is that it does not give you discrete output or outright classes as output, but instead you get probabilities associated with each observation'[27]. This probability is then converted to one of two classes, depending on the threshold.

If there is an association with a feature and a certain outcome, this association might be missed if the data set is overly saturated with raw, useless data[28]. This is important to keep in mind when pre-processing data and extracting features, as we can draw more concise conclusions when less features are at play.

3.1.6 *Linear Discriminant Analysis (LDA)*

Linear Discriminant Analysis (LDA), as stated by Tharwat et al. [29], tries to 'project the original data matrix onto a lower dimensional space.' It does this while retaining the information needed to differentiate instances into classes.[30]

One of the main drawbacks of LDA is known as the Small Sample Size problem: it 'fails to find the lower dimensional space if the dimensions are much higher than the number of samples in the data matrix.'[29] We will be using a sizable sample of control data, however the amount of instances that belong to the 'attack' class are far and few between, which might drastically hamper performance.

3.2 Findings

In the previous section we found that all cloud attacks in Table 1 have at least a classification problem type. For this reason we will be using the classifiers detailed in this section for monitoring cloud attacks, as this covers the most ground for our research. It enables future works to easily expand on the foundations laid by this research.

However, there were some attack techniques that could be tackled using a different approach. When it is not clear what you are looking for, e.g. labels and attributes/features, it might be beneficial to use an unsupervised ML algorithm, e.g. k-means. For example, in the case of detecting malicious URLs (TI-3), Sahoo et al. [15] state that "the number of URLs available for training can be in the order of millions (or even billions) and as a result, the training time for traditional models may be too high to be practical". Thus, in some scenarios, a clustering approach might be more suitable than a classification approach. This could be explored in future works.

To know which classifier performs best for monitoring a given attack, the best course of action is to conduct a performance evaluation on different ML algorithms. We detail this process in the next section, where we use metrics to decide which classifier best suits the monitoring of the cloud attacks that are highlighted in Table 1.

4. PERFORMANCE EVALUATION

The intent of this section is to provide the methodology of conducting a performance evaluation on the ML algorithms that were discussed in the previous section. The end of this section includes our findings.

4.1 Methodology

To answer RQ3 (*How well does ML perform when monitoring cloud attacks?*) we have conducted a performance evaluation of ML algorithms for monitoring the two cloud attacks (TI-5) 'Valid Accounts' and (TI-15) 'Brute Force' from Table 1. The exact methodology, including the tools used, is described in distinct subsections below.

4.2 Tools Used

In this subsection we will go over the tools used for the performance evaluation. First of all, we used Jupyter Notebook[31] using Python. Jupyter Notebook is an open-source web-application for executing live-code. Not only is this very useful for visualizing the results, but also enables the reproducibility of our work as the source code is publicly available on GitHub: `https://github.com/NilsvN/my_thesis`

The ML algorithms we utilized all come from a Python module called Scikit-learn (SKLearn)[32]. This module contains many supervised and unsupervised ML algorithms. This module helped with the time it took to compare the performance of in total 6 ML algorithms for each attack.

We used Microsoft Azure[33] for simulating attacks. The two cloud attacks that we will evaluate ML on (TI-5, TI-15, Table 1) are simulated on Azure by mimicking the actions an actual attacker would go through. We will detail this process along with its drawbacks in the subsections below. First though, we will mention one other tool used in a separate subsection.

4.3 Dataset and its Anonymization

The two highlighted attacks are both regarding account sign-ins, so we needed to acquire 'normal' user sign-in logs. To this end, we collaborated with Northwave[34] to develop a tool that is able to automatically anonymize the sign-in logs[35] from Azure Log Analytics. After assigning which columns contain sensitive (personally identifiable) information, the sensitive entries will be systematically substituted for anonymous data. Here, context is preserved, e.g. username A is be replaced by anonymous username B at every instance of A.

This tool is also made using Jupyter Notebook and Python and is open to the public. It can be very useful for problems similar to what is faced in this research. In fact, any future works can use this tool to decrease the time it takes to obtain the necessary data. The usefulness of this tool warranted its own subsection. Now that we have covered the tools used for performance evaluation, we will detail how we utilized these tools.

4.4 Datasets

The first step of the performance evaluation entails gathering the datasets. Using the tool from § 4.3 we extracted 30 days of anonymized Azure sign-in logs. These logs were downloaded and saved as *.csv* so we could later import it into our Jupyter Notebook environment.

Gathering the data for the two attacks involved more manual interactions. As we did not have data on actual attacks, it was more beneficial to mimic the attacks by going through the steps an attacker would go through. We first created a new Azure tenant so we could simulate attacks in a closed environment. This simulated approach definitely has its drawbacks, which we will discuss in § 4.6. The rest of this subsection will detail the aspects to look for when monitoring each attack, along with how we replicated certain aspects of the attack. Finally we mention how we extracted the attack logs.

4.4.1 (TI-5) 'Valid Accounts'

The 'Valid Accounts' attack involves accounts that have already been compromised, which pose a serious threat to any system. We found there to be two main aspects to look for when monitoring such an attack.

Firstly, perhaps the most obvious one is to look for account sign-ins from different locations in a short time span. This is a strong indication that an account might be compromised. For example, two successful sign-ins within an hour of each other but from different locations in the world are a good candidate. For our simulated attack, we used a VPN to sign in from the USA and a few minutes later sign into the same account from the Netherlands.

Secondly, we look for multiple account sign-ins from a single IP address. While it is true that some users often switch between multiple accounts, the chance that in such a scenario one of the accounts is compromised is higher. This was simulated by signing into two Azure accounts within minutes of each other from the same machine. As an example, attackers might share a compromised account, which means there is a good chance that both the mentioned aspects are relevant.

4.4.2 (TI-15) 'Brute Force'

'Brute Force' is an attack that involves an attacker trying passwords on one or multiple accounts repeatedly, in hopes of signing in successfully by chance. Here, coincidentally, there are also two main ways an attacker could go about performing such an attack.

First of all the attacker could try logging in to a single account many times. The amount of failed sign-in attempts is the main thing to look for when monitoring this attack. On Azure, there are already systems in place to block an account from sign-ins after a certain amount of failed attempts. This did not stop us from trying to log in 20 times in a row for simulating the attack.

The other aspect of the attack is what is known as 'password spraying'. This involves the attacker trying a limited amount of commonly used password on many different accounts. This method is smarter than the one above, as you might avoid simple account blocks for too many failed attempts. To simulate this we created 10 different Azure accounts that we each tried to log into once.

To end this step in the methodology, we extracted the variants of each attack from the 'Logs' page of the Azure tenant by downloading all columns from the 'SigninLogs' table. Since we have two variants of two attacks we extracted 4 *.csv* files in total that were filtered to only include the attacks. All datasets are publicly available here: https://github.com/NilsvN/my_thesis/tree/master/datasets

4.5 Feature extraction

The second major step entails the extraction of features from the datasets. The 'Brute Force' and 'Valid Accounts' attack both have a dedicated *.ipynb* Jupyter Notebook file. The difference lies in the functions created for feature extraction. Before going into the feature extraction functions, we must first prepare the data. We will briefly describe this process.

For each variant of an attack there is a training and testing version, meant to minimize the possibility of overfitting the ML. The difference being a change in sign-in attempts and/or an altered date/year. The training sets of each attack are concatenated into one, and then combined with a subset of the 30 day sign-in logs. The same is done for the testing sets. All the while, the attack logs are labelled as '1' while the normal 30-day logs are labelled as '0' to suit the classification algorithms. Finally, each individual log contains a column 'TimeGenerated' which displays the date and time that this log was generated. This column is converted from date/time string to Unix time (seconds since January 1, 1970, 00:00:00 UTC).

It is obvious that the final datasets are very much imbalanced. Instead of trying to balance this dataset by either up-sampling or down-sampling, we chose to embrace the imbalance. If we were to balance the data, the performance of the classifier in the real world would be lower because the actual proportions of classes was not represented well during training[36].

The sign-in logs contain many columns containing irrelevant data for monitoring a specific attack. This is noise that can hamper overall performance, so the data requires a significant amount of pre-processing to get the most relevant features[37]. For each attack, we describe the unique features we extracted from the logs and why those are most relevant to monitoring the attack.

4.5.1 (TI-5) 'Valid Accounts'

For monitoring the 'Valid Accounts' attack we will keep in mind the two aspects that could point towards an account being compromised. As such the functions put in place to extract useful features for the ML have the following functionalities:

- For each row, save the amount of different locations from which this account was successfully logged into within a certain time window (`loc_delta_s`)
- For each row, save the amount of different accounts this IP successfully logged into within a certain time window (`delta_s`)

Here, `delta_s` and `loc_delta_s` represent the difference between the log Unix time and the lower/upper bound of the time window (in seconds), which can be tweaked independently.

4.5.2 (TI-15) 'Brute Force'

For monitoring the 'Brute Force' attack we will keep in mind the two approaches an attacker could use, either trying to force passwords on a single account or trying a common password on many accounts. The functions put in place to extract useful features for the ML have the following functionalities:

- For each row, save the amount of times this machine attempted to sign into this account within a certain window (`delta_s`)

- For each row, save the amount of times this IP tried to sign into any account (`ip_delta_s`)
- For each row, save the amount of different accounts this IP tried to log into within a certain time window (`ip_delta_s`)

Same as before, `delta_s` and `ip_delta_s` represent the difference between the log Unix time and the lower/upper bound of the time window (in seconds), which can be tweaked independently.

Now that we have covered the attack specific features we can look at the last feature that is present for both attacks, namely the sign-in status. This status is either a successful login ('success') or an unsuccessful login ('failure'/'notApplied'). This is a nominal categorical variable, meaning the different statuses have the same weight (in contrast to e.g. sizes M/L/XL). In order to effectively include this categorical feature along with the other, value-driven features we used 'One-Hot Encoding'[38]. This replaces the original column with X amount of columns, where X is the amount of different categories. Each column will be filled with a '0' or a '1', depending on which category this row is in.

4.6 Algorithms and Thresholds

The next part of the methodology entails using ML classification algorithms from SKLearn[32]. We will describe how we implemented a `threshold` variable that should allow for optimized results. Finally we end this subsection by detailing the drawbacks associated with our approach.

The classification algorithms we used are the 6 ML algorithms that we detailed in section § 3. SKLearn offers a straightforward way to utilize these algorithms. For each ML algorithm, we implemented an option to include a `threshold` variable. This variable replaces the standard value of 0.5 when assigning a label. For example, if the algorithm finds that the probability that a certain entry belongs to class '1' is equal to 0.35, it would normally output class '0'. However, if we set `threshold` to <0.35 it predicts class '1'. When testing we used an interval of [0, 1] with 100 total thresholds evenly spread throughout. The exception is k-Nearest Neighbours, which instead takes an integer `k` as variable. Naturally, instead of an interval of [0, 1] we used an interval [1, 50] with 50 total values (1, 2, . . . , 50).

It is important to realize the drawbacks of our approach that involves mimicking the attacks. The danger of such an approach is that the tests we have conducted might not represent a real world scenario well. This could lead to conclusions drawn that do not match the real performance. Another aspect to worry about is the concept of overfitting. There is a certain risk that the algorithm is overfitted to match the mimicked attack, such that this algorithm would not perform well in a real environment. We tried to mitigate this by manually editing the attacks so that the training data does not match the testing data in any meaningful way.

4.7 Evaluation Metrics

As the next step in our methodology we cover the definitions of metrics used to measure performance. Typically, the performance of a classifier is presented using a confusion matrix[39]. This matrix shows the amount of erroneous and successful predictions; as such a sign-in log can be classified into one of the four following groups:

- True Positive (T_P): When the classifier correctly predicts an attack log as 'attack'.
- False Positive (F_P): When the classifier incorrectly predicts a normal log as 'attack'.
- True Negative (T_N): When the classifier correctly predicts a normal log as 'normal'.
- False Negative (F_N): When the classifier incorrectly predicts an attack log as 'normal'.

The accuracy (ACC) of a classifier is given by

$$ACC = (T_P + T_N)/n$$

where n is the total number of logs.

The precision (PRC) and recall (REC) of a classifier are defined as follows, respectively:

$$PRC = T_P/(T_P + F_P)$$

$$REC = T_P/(T_P + F_N)$$

These metrics are used to express the performance of the classifiers for each attack. In the next subsection we will analyze the performance where the metrics are plotted against the threshold.

4.8 Evaluating Performance

As the last step, we will analyze the plots in Figure 1 and Figure 2 displayed on the next page. We will split this analysis into two parts for the respective attacks.

4.8.1 (TI-5) 'Valid Accounts'

When looking at the plotted metrics, we can see that there is no classifier that at any point reaches a value of 1.0 for accuracy, precision and recall simultaneously. Many of them do have a high accuracy though (all except (b)), however this does not draw the complete picture. For that we must also look at the precision (PRC) and recall (REC).

Besides the accuracy, one could argue that the most important metric is the recall. The reasoning being that is that you would prefer to be in a situation of a false alarm than a situation where you missed a breached account. As such a F_N could do much more damage in a real environment than a F_P. This is important to keep in mind when judging which algorithm(s) performed the best.

With this in mind, we summarized the classifiers that performed best along with their optimal `threshold`/`k` range:

- (a) SVM ($0.01 \leq$ `threshold` ≤ 0.99)
- (c) K-NN ($1 \leq$ `k` ≤ 7)
- (d) Decision Trees ($0.01 \leq$ `threshold` ≤ 0.99)

The reason some of the plots show rather straight lines is because the datasets are imbalanced, as previously mentioned. This means that any change in F_P or F_N makes a big difference for the metrics shown. Also, the fact that the best performers, (a), (c), and (d), all max out at a recall of 0.67 has one main reason. One of the variants of the 'Valid Accounts' attack was not predicted correctly by any of them.

When analyzing further, the features that make this attack suspicious can occur in a similar fashion within normal sign-in logs. For example, there are plenty of users that switch between 2 or more accounts often, or those that use a VPN. This is exactly why you sometimes receive an e-mail along the lines of 'Was this you? You signed in from country X'. You simply cannot be certain if the account is compromised or if this was a regular user action, which means this is a limitation of traditional algorithms as well.

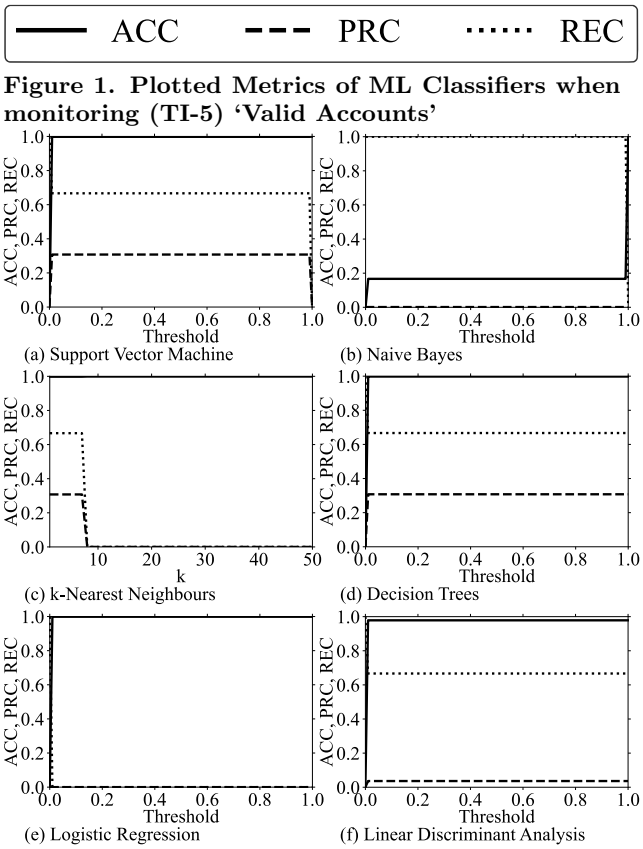

Figure 1. Plotted Metrics of ML Classifiers when monitoring (TI-5) 'Valid Accounts'

We found best results when `delta_s = 120` (seconds) and `loc_delta_s = 1800` (seconds, equal to 30 minutes)

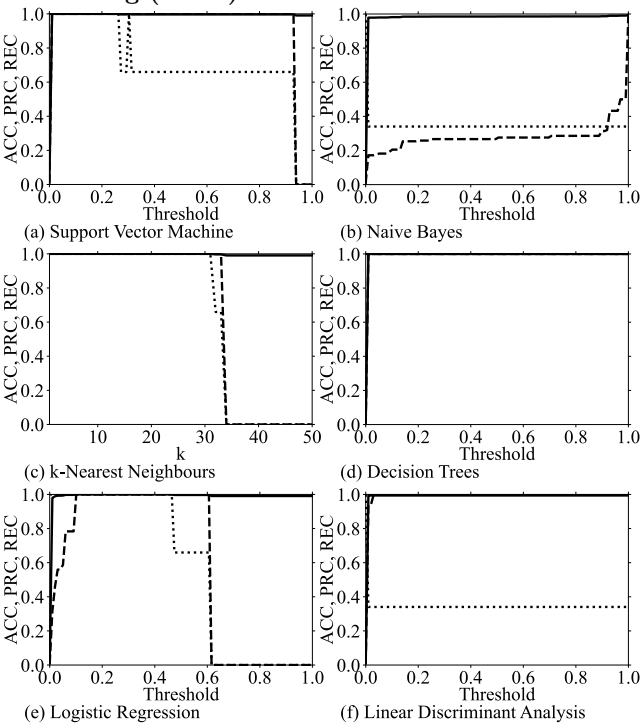

Figure 2. Plotted Metrics of ML Classifiers when monitoring (TI-15) 'Brute Force'

We found best results when `delta_s = 120` (seconds) and `ip_delta_s = 120` (seconds)

4.8.2 *(TI-15) 'Brute Force'*

For this attack, we can see quite a difference in results when comparing the plots to the previous ones. Here, the maximum metrics of 1.0 for accuracy, precision and recall is reached by multiple classifiers. It is safe to say that the features extracted for monitoring this attack showed enough difference between normal logs and attack logs that the classifiers could predict more accurately.

Once more for this attack, we summarized the classifiers that performed best with their optimal `threshold`/`k` range:

- (a) SVM ($0.01 \leq$ `threshold` ≤ 0.26)
- (c) K-NN ($1 \leq$ `k` ≤ 31)
- (d) Decision Trees ($0.01 \leq$ `threshold` ≤ 0.99)
- (e) Logistic Regression ($0.10 \leq$ `threshold` ≤ 0.46)

The above classifiers all managed the maximum metrics of 1.0, which means no erroneous predictions. This is quite a promising result when you consider that both variations of the 'Brute Force' attack are classified with 100% accuracy. The results from both (TI-5) and (TI-15) are presented in what comprises the start of an overview that shows exactly what ML algorithm to use for which attack, Table 3 below.

5. CONCLUSION

Cloud and ML are both becoming increasingly popular. However they seem to be doing this mostly separated from one another. Most existing works are merely scratching the surface when it comes to ML capabilities in a cloud environment, as they often use ML for a small subset of attacks. This results in an information gap which raises uncertainty whether or not ML is a worthwhile investment for monitoring cloud attacks. We proposed a methodology in order to pave the way for future research on using ML for monitoring cloud attacks. For ease of reproducibility and expansibility, the source code is publicly available here: https://github.com/NilsvN/my_thesis

While we witnessed promising results, it is important to note that the trickiest part of utilizing ML for another attack is likely to be the feature extraction, depending on the nature of the attack. Our methodology thoroughly covers all steps involved in utilizing ML; from acquiring the datasets, to feature extraction, to threshold optimization. We hope this incentivizes further research on this topic.

In future works, we aim to expand the overview of Table 3 to include an increasing number of attacks for which the performance of ML algorithms has been evaluated. Beside this horizontal development, future works may also look at further optimizing the ML algorithms used. For example, a 'Cost-based classification' approach might be beneficial, as a F_N carries more weight than a F_P.

With this research, we aim to incentivize further development on using ML for monitoring cloud attacks. ML has shown promising results when used against the attacks covered in this paper. While traditional algorithms might perform similarly to what we found ML capable of, there are other cloud attacks of which the high-dimensionality is likely to favour ML over a traditional approach. We believe such a possible prospect could warrant further research to ultimately provide a full overview of exactly which ML algorithm to use for which attack.

Table 3. Overview of Classifiers & Techniques

TI	Technique	Tactic	SVM	Naive Bayes	k-Nearest Neighbours	Decision Trees	Logistic Regression	Linear Discr. Analysis
5	Valid Accounts	IA, P, PE, DE	✓	✗	✓	✓	✗	✗
15	Brute Force	CA	✓	✗	✓	✓	✓	✗

IA: Initial Access, P: Persistence, PE: Privilege Escalation, DE: Defence Evasion, CA: Credential Access

Acknowledgement

We would like to thank J.J. Cardoso de Santanna from Northwave (https://northwave-security.com/) for collaborating with us on the tool for anonymizing Azure sign-in logs, as this tool turned out to be invaluable to our research and undoubtedly so to future works as well.

References

[1] Gartner. *Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.5 Percent in 2019*, 2019 (accessed April 30, 2020). https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g.

[2] Cybersecurity Insiders. *2019 Cloud Security Report - ISC2*, 2019 (accessed June 27, 2020). https://www.cybersecurity-insiders.com/wp-content/uploads/2019/05/2019-Cloud-Security-Report_ISC2_1.7.pdf.

[3] UpGuard. *Losing Face: Two More Cases of Third-Party Facebook App Data Exposure*, 2019 (accessed May 1, 2020). https://www.upguard.com/breaches/facebook-user-data-leak.

[4] MITRE Corp. *Web Session Cookie*, 2020 (accessed May 1, 2020). https://attack.mitre.org/techniques/T1506/.

[5] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[6] NCS Analytics Inc. J. Adams. *Buzzwords & Bullsh!t: Machine Learning*, 2018 (accessed May 2, 2020). https://www.ncsanalytics.com/media/buzzwords-and-bullsht-machine-learning/.

[7] Index.co. *Machine Learning Investors*, 2020 (accessed May 3, 2020). https://index.co/market/machine-learning/investors.

[8] Marleen De Bruijne. Machine learning approaches in medical image analysis: From detection to diagnosis, 2016.

[9] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[10] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Syst. Appl.*, 36:11994–12000, 2009.

[11] Diogo Fernandes, Liliana Soares, João Gomes, Mario Freire, and Pedro Inácio. Security issues in cloud environments - a survey. *Int. J. Inf. Secur.: Security in Cloud Computing*, 2013.

[12] MITRE Corp. *MITRE ATT&CK*, 2020 (accessed May 1, 2020). https://attack.mitre.org/.

[13] MITRE Corp. *Cloud Matrix*, 2020 (accessed May 1, 2020). https://attack.mitre.org/matrices/enterprise/cloud/.

[14] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[15] Doyen Sahoo, Chenghao Liu, and Steven CH Hoi. Malicious url detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179*, 2017.

[16] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, 1(1): 4–20, 2010.

[17] Andrew W Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, 2005.

[18] Uday Trivedi and Munal Patel. A fully automated deep packet inspection verification system with machine learning. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2016.

[19] Phillip George Efthimion, Scott Payne, and Nicholas Proferes. Supervised machine learning bot detection techniques to identify social twitter bots. *SMU Data Science Review*, 1(2):5, 2018.

[20] Shailendra Rathore, Pradip Kumar Sharma, and Jong Hyuk Park. Xssclassifier: An efficient xss attack detection approach based on machine learning classifier on snss. *JIPS*, 13(4):1014–1028, 2017.

[21] Solomon Ogbomon Uwagbole, William J Buchanan, and Lu Fan. Applied machine learning predictive analytics to sql injection attack detection and prevention. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1087–1090. IEEE, 2017.

[22] Sotiris Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31, 2007.

[23] Giulio D'Agostini. A multidimensional unfolding method based on bayes' theorem. Technical report, P00024378, 1994.

[24] S. Robinson. *K-Nearest Neighbors Algorithm in Python and Scikit-Learn*, 2018 (accessed June 15, 2020). https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/.

[25] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[26] Leonard A Breslow and David W Aha. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, 12(01):1–40, 1997.

[27] A. Shah. *Logistic Regression vs Decision Trees vs SVM: Part II*, 2015 (accessed June 15, 2020). https://www.datasciencecentral.com/profiles/blogs/logistic-regression-vs-decision-trees-vs-svm-part-ii.

[28] S. Sperandei. *Understanding logistic regression analysis*, 2014 (accessed June 10, 2020). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3936971/.

[29] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien. Linear discriminant analysis: A detailed tutorial. *AI communications*, 30(2): 169–190, 2017.

[30] U. Malik. *Implementing LDA in Python with Scikit-Learn*, 2018 (accessed June 13, 2020). https://stackabuse.com/implementing-lda-in-python-with-scikit-learn/.

[31] Project Jupyter. *Jupyter Notebook*, 2020 (accessed May 2, 2020). https://jupyter.org/.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] Microsoft. *Microsoft Azure*, 2020 (accessed June 18, 2020). https://azure.microsoft.com/en-us/.

[34] Northwave. *Northwave Intelligent Security Operations*, 2020 (accessed June 21, 2020). https://northwave-security.com/.

[35] J.J. Cardoso de Santanna. *anonymising-azure-sign-in-logs*, 2020 (accessed June 12, 2020). https://github.com/jjsantanna/anonymising-azure-sign-in-logs/blob/master/anonymising-azure-sign-in-logs.ipynb.

[36] B. Rocca. *Handling imbalanced datasets in machine learning*, 2019 (accessed June 12, 2020). https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28.

[37] Qiang Yang Shichao Zhang, Chengqi Zhang. Data preparation for data mining. *Applied Artificial Intelligence*, 17(5-6):375–381, 2003.

[38] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. Similarity encoding for learning with dirty categorical variables. *Machine Learning*, 107(8-10):1477–1494, 2018.

[39] Mohamed Hammami, Youssef Chahir, and Liming Chen. Webguard: A web filtering engine combining textual, structural, and visual content-based analysis. *IEEE Transactions on Knowledge and Data Engineering*, 18(2):272–284, 2005.