CMD Obfuscation Detection Tool

Research Document

Student: Melanie Dudas
Student Number: C00253245
Supervisor: Dr Hisain Elshaafi

Table of Contents

Table of Figures

Abstract

In cybersecurity, threat detection and response are among the most important things if damage and disruption of a system are to be minimised. Detecting a potential compromise in its early stages is critical to limit the damage and stop the malicious actor from breaching even further. Sometimes, even the best traditional detection systems are not capable of detecting malicious activities, and that is where Machine Learning can help because it can automate the process and make it more effective.

This research document is part of a project focusing on a recently discovered threat capability: command line obfuscation. This project aims to gather an understanding of the threat and to learn how to use Machine Learning as a solution for it effectively. This research document investigates existing and potential techniques that can be used to detect the exploitation of command line obfuscation, including machine learning based approaches. The research shows that Gradient Boosting model consistently had the best accuracy measures for obfuscation detection.

Introduction

CMD, CMD.exe, Command Prompt or Windows Command Processor is a Windows command line interpreter application used to execute commands. CMD's role as an interpreter means it translates the commands users give into machine language to allow users to interact with the Operating System.

Daniel Bohannon researched CMD obfuscation in-depth and presented his findings at Black Hat Asia 2018 and more detailed in DOSfuscation White Paper (Bohannon, 2018). Three years later, Wietze Beukema furthered the research even more and came up with the term Synonymous Command Lines and five methods of how they could be used (Beukema, n.d.). The difference between DOSfuscation and Synonymous Command Lines is that the obfuscation is no longer tricking just the CMD but the executing program as well. The biggest problem synonymous command lines present to security is that, unlike DOSfuscation methods, synonymous command line arguments do not present unobfuscated in the recorders used by detection software. For that reason, this research document focuses on synonymous command lines.

To form the best solution for the security problem companies worldwide are facing, this research document initially examines the threat in depth. It explains why malicious actors benefit from CMD obfuscation. Afterwards, the limitations of commonly used static detections are discussed. Additionally, this research document focuses on a machine learning solution and currently available machine learning approaches for the problem at hand. Finally, the technologies used to develop a new machine learning solution are investigated.

CMD Obfuscation

CMD does not have a set of syntax or character encoding rules which tools running from CMD need to follow. For example, most Windows-native command-line tools use forward slashes for their option character, unlike in Unix systems where the standard practice is to use a hyphen. To avoid unsuccessful execution, some programs accept hyphens and slashes, while others go even further and accept most Unicode representations of slashes. Figure 1 shows testing netstat.exe by changing the option characters with several alternatives and the finding was that the program allowed eight different ones (0x002F, 0x2010, 0x2012, 0x2015, 0x2044, 0x2212, 0x2215 and 0xFF0D).

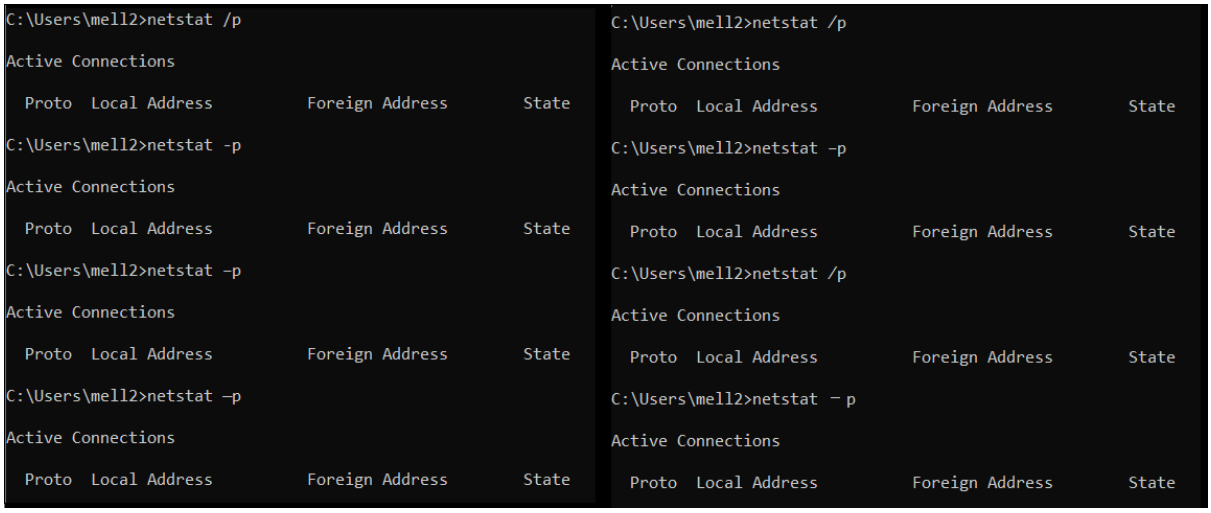

Figure 1 *Option char substitution example*

A problem similar to this one arises from different character encodings. Most older programs accept only ASCII characters, while the newer ones accept UTF-8 or Unicode. To "fix" the issue, some programs try to find the ASCII equivalent and convert the character into that, and some ignore certain characters. Below, netstat.exe was tested again, but this time for character insertion. The program worked the same even when the letter p – 0x50 was substituted with 0x03C0, 0x20A7, 0x2118, 0x2119, 0xFF30 or 0xFF50.

Figure 2 *Character substitution example*

Because netstat.exe allowed for character substitution, character insertion was also tested, and the program successfully ignored the inserted characters. The inserted characters, shown in the screenshot below, were 0x03A6, 0x20A7, 0x221A, 0x2229, 0x0147 and 0x03C60.

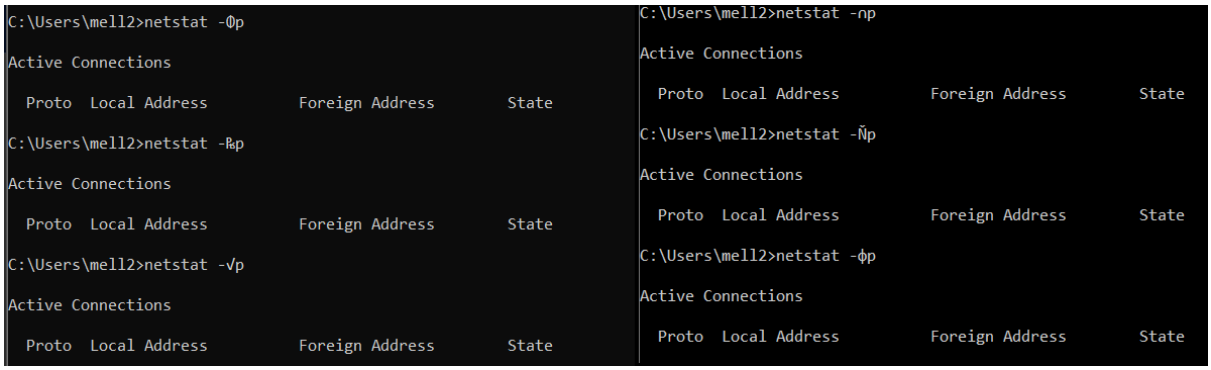

Figure 3 *Character insertion example*

Regarding quotes, most programs will allow quote insertion and ignore the quotes if they come in pairs. Below, powershell.exe was tested with quotes inserted and the program worked as if there were no quotes.

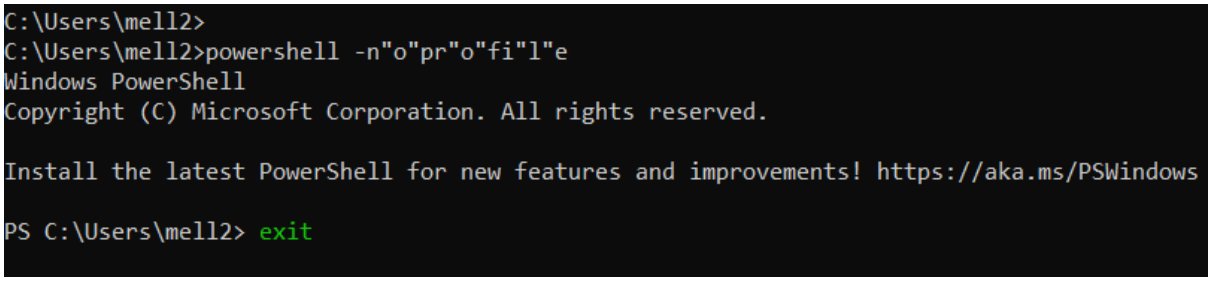

Figure 4 *Quote insertion example*

Some programs also allow removing certain characters. Windows is not that well-known for shortening commands compared to Unix systems. Depending on the program, the command can be shortened to only one letter or another abbreviated version. Some programs are even more flexible, so PowerShell accepts any shortened version as long as it does not cause ambiguity between another command. While shortening a command to only one letter makes sense because it saves the user's time, allowing the command to be shortened by removing only one or two characters at the end only makes writing the detections more complicated. (Beukema, n.d.)

Why Obfuscate CMD

Why would someone add all those extra steps to do the same thing that would happen without the obfuscation? The answer is – to avoid being detected for as long as possible. For a company to respond to an attack, it first needs to recognise that it has been attacked. Detecting a security breach can take months or even years. One of the studies focusing on security breaches found that the average time to identify a security breach in 2022 was 207 days. (Tunggal, 2022)

Evading detection by using obfuscation is a well-known malware tactic. Malware can be obfuscated in many ways, such as encoding, encryption, compression, packing, string manipulation etc. It can be challenging to detect if the obfuscation of a program is malicious or if it was done for a legitimate reason, like using the packing technique to protect the application's trade secrets.

On the other hand, there is no legitimate purpose for obfuscating CMD, which makes any obfuscated command a suspicious activity that must be investigated as soon as possible. Obfuscating a command in CMD does not guarantee that the malicious activity will not be detected in another way. Using a program that is suspicious by itself is enough for the company's static detections or EDR to flag the activity as suspicious and stop it. That is why CMD obfuscation is often combined with Living off the Land Binaries.

Living off the Land Binaries

Living off the Land Binaries or LOLBins are trusted local tools that come pre-installed as a part of the operating system. These tools are used for legitimate purposes but have extra capabilities that malicious actors can exploit.

A great example of a LOLBin would be CertUtil.exe. It is a Windows binary used for handling certificates. Its capabilities include the following:

- Download - can be used to download malicious files from a given URL - MITRE ATT&CK®: T1105
- Alternate data streams – can be used to download a file from the Internet and save it in an NTFS Alternate Data Stream - MITRE ATT&CK®: T1564.004
- Encoding - can be used to encode files to evade defensive measures - MITRE ATT&CK®: T1027
- Decoding – can be used to decode files to evade defensive measures - MITRE ATT&CK®: T1140

(Moe, et al., n.d.)

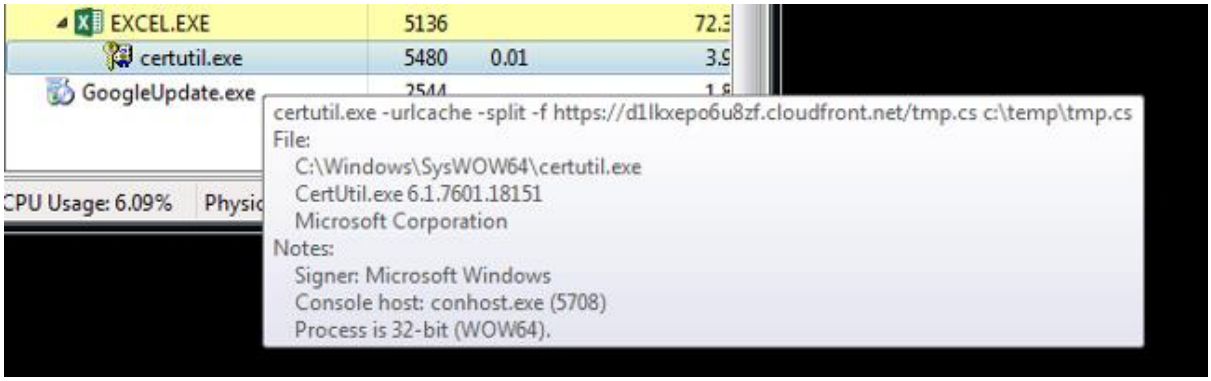

Figure 5 A real-life example CertUtil used maliciously. (Avira Protection Labs, 2020)

Living off the Land Binaries is not a new term, and security teams are aware of how often tools like that are exploited, so having detections in place for them is a standard security practice. Because the tools are legitimate, they are often used for legitimate purposes, so the detections can be very noisy. That means the detections have to be written so that they are not too general, as that would set off too many false-positive alarms, but also not too specific so that they do not miss the malicious activity.

Writing the detections can be tricky, but if we take into account CMD obfuscation, tricky becomes impossible.

Rule-Based Detections

Detections for CMD are written mainly by looking for cmd.exe with specific arguments, process execution events, or parent-child process relationships. Although rarely, detections could be set in place that are looking for cmd.exe as a source of action, for example, creating/modifying a file or setting a registry run key.

In most cases, detections are written with Sigma rules. Sigma rules detect anomalies in an environment by monitoring log events and looking for signs of suspicious activity and threats. (Sharma, 2022) To detect potential malicious use of a binary, a Sigma rule, looking for an execution of that binary with a specific command-line argument, would be written.

If we wanted a detection for the above example of CertUtil, we would write a Sigma rule which is looking for CertUtil execution with -urlcache and -f arguments.

```
detection:
    selection:
        Image: "*\\certutil.exe"
        CommandLine|contains|all:
            - "/urlcache"
            - "/f"
    condition: selection
```

Figure 6 *Example of a Sigma rule for CertUtil*

However, that rule would not detect the malicious use of CertUtil if the command was even slightly obfuscated. Had the option char been replaced with a hyphen:

/f ⇨ -f

The command would work, but we would not get a detection for it. To solve that, we could use regex or regular expressions. (Beukema, n.d.) Regular expressions are patterns used to match character combinations in strings. (MDN, 2022)

```
detection:
    selection:
        Image: "*\\certutil.exe"
        CommandLine|re: "^(?=.*[-/\\]urlcache)(?=.*[-/\\]f).+.*"
    condition: selection
```

Figure 7 *Sigma rule for CertUtil using regular expressions*

The regex example above solves the option character substitution problem, but what about the rest of the obfuscation techniques? What if the command has a special character inserted somewhere in the middle or if the letter L is changed to ʟ?

The command could then look like this:

- certutil – urLcache
- certutil – ⚘urlcache
- certutil – u”r”lca”c”he

Or even all those things combined:

➢ certutil – ⚘u”r🜲”Lc⃠a”c”h✧e

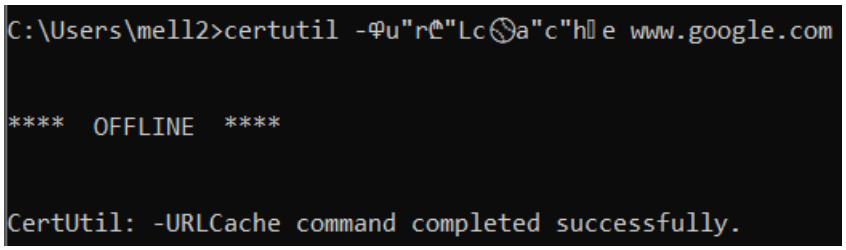

Figure 8 *Successful execution of CertUtil despite obfuscation*

One could argue that regex could still be used to try and catch the obfuscation, and sure, it would be easy to use regex to cover all possible option char substitutions. Then the problem is to try and include all quotes that can be inserted anywhere as long as there are an even number of them. If that is also managed, the next step is to include all other possible characters that can be inserted into the command. An important thing to note here is that CertUtil has 1758 known characters that can be inserted anywhere. (Beukema, n.d.) This is where the regex becomes so complex that not only would it be complicated to create it, but even if it’s created, it would probably lead to a never-ending evaluation of the expression.

Machine Learning

Machine Learning is a subfield of Artificial Intelligence that allows computer programs to predict outcomes without being explicitly programmed to do so. (Burns, 2021) A machine learning algorithm is an algorithm that can learn from data. Tom M. Mitchell, in Machine Learning book (Mitchell, 1997), defines it well: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*".

If Machine Learning was used to detect the obfuscated command line, the task *T* would be for the computer program to specify in which of the categories, obfuscated or not obfuscated, the input belongs, i.e., to classify it. The performance measure *P* is a quantitative measure of the algorithm's performance. It is usually specific to the task *T* being carried out by the system. For the above task *T,* precision, recall and f1 of the model would be suitable performance measures. Accuracy measure could be used as it is simply the proportion of examples for which the model produces the correct output. However, accuracy would not be a good performance measure for the problem at hand simply because the dataset in real life is imbalanced. Only a fraction of commands are obfuscated, so even if the model predicts everything as not-obfuscated, it would still be 99% accurate. That is why precision, recall and f1 are needed. Precision measures how often the positive prediction is correct, meaning how often the "obfuscated" command is really obfuscated. It is measured as follows:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall measures what proportion of actual positives was identified correctly, meaning how often a truly obfuscated command was predicted as “obfuscated”. It is measured as follows:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The F1 score measures the accuracy by combining both precision and recall. That would mean having an excellent f1 score, as close to f1=1 as possible, indicates the program has a low number of false positives and false negatives. It is measured as follows:

$$f1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

(Nicholson, n.d.)

Lastly, experience *E* ties to what kind of experience, supervised or unsupervised, the algorithm is allowed to have during the learning process. The learning algorithms are experiencing a collection of many examples called a dataset or data points. In supervised learning, an "instructor" shows the machine learning system what to do, while in unsupervised learning there is no "instructor" and the algorithm needs to learn to make sense of data by itself. An unsupervised learning algorithm experiences a dataset, random vectors *x*, and attempts to learn the probability distribution, *p(x)*. Supervised learning algorithm also experiences a dataset, random vectors *x*, but with an associated vector *y* and then it learns to predict *y* from *x*, usually by estimating *p(y|x)*. (Goodfellow, Bengio, & Courville, 2017)

The main worry about the Machine Learning approach is the accuracy of the program. IBM's annual Cost of a Data Breach Report shows that the average cost of a data breach in 2022 is USD 4.35 million. (IBM, 2022) Suppose the company entirely relies on the machine learning approach and the program misses even one obfuscated, malicious use of CMD. In that case, that is how much money the company might lose, not including the damage to the brand's reputation. However, in cybersecurity, it is impossible to detect everything malicious that will ever happen. There will always be something new the malicious actors think of, so having the most detections as possible is crucial. It's better to have a machine learning program that only detects a portion of obfuscated CMD than to have nothing in place that is on the lookout for that particular problem.

Machine Learning Algorithms

A machine learning algorithm can be defined as an algorithm capable of improving a software application's performance of a task by its experience. The algorithm must be accurate in its performance on new, not yet seen inputs, not only on the inputs on which the model was trained. In this section, four different algorithms will be researched and explained.

Convolutional Neural Network

Mandiant and Adobe used the Convolutional Neural Network algorithm to develop a machine learning detector of CMD obfuscation. Although the primary usage of this algorithm is for analysing visual imagery, it has shown promising results in natural language processing (NLP). (Hedge, 2018)

Convolutional Networks are Neutral Networks that use convolution in place of general matrix multiplication in at least one of their layers. (Goodfellow, Bengio, & Courville, 2017) Convolution is a mathematical operation on two functions that produces a third function which expresses how the

shape of one is modified by the other. A CNN model has three layers: a convolutional layer, a pooling layer and a fully connected layer. (Mandal, 2021)

The convolution layer is where the majority of computations happen, and there can be more than one convolution layer.

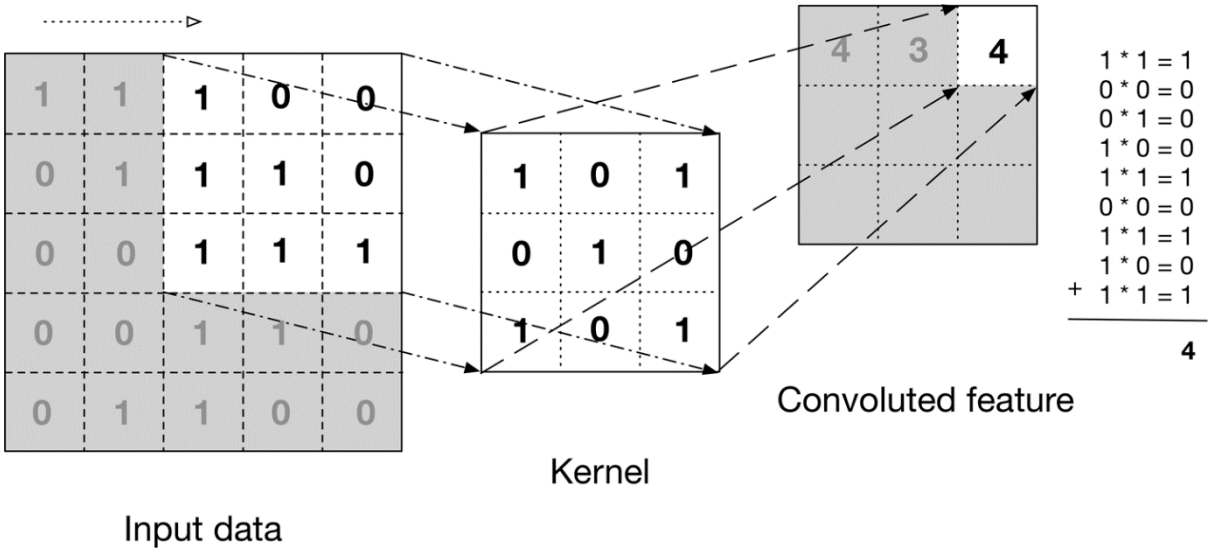

Figure 9 *Convolution (Mandal, 2021)*

The pooling layer is used to reduce the spatial size of the Convolved Feature to decrease the computational power required to process the data. (Mandal, 2021) Two types of pooling can be used: Max Pooling and Average Pooling. The difference between the two is straightforward, Max Pooling finds and takes the maximum value of every pool while Average Pooling gets the average value of the pool.

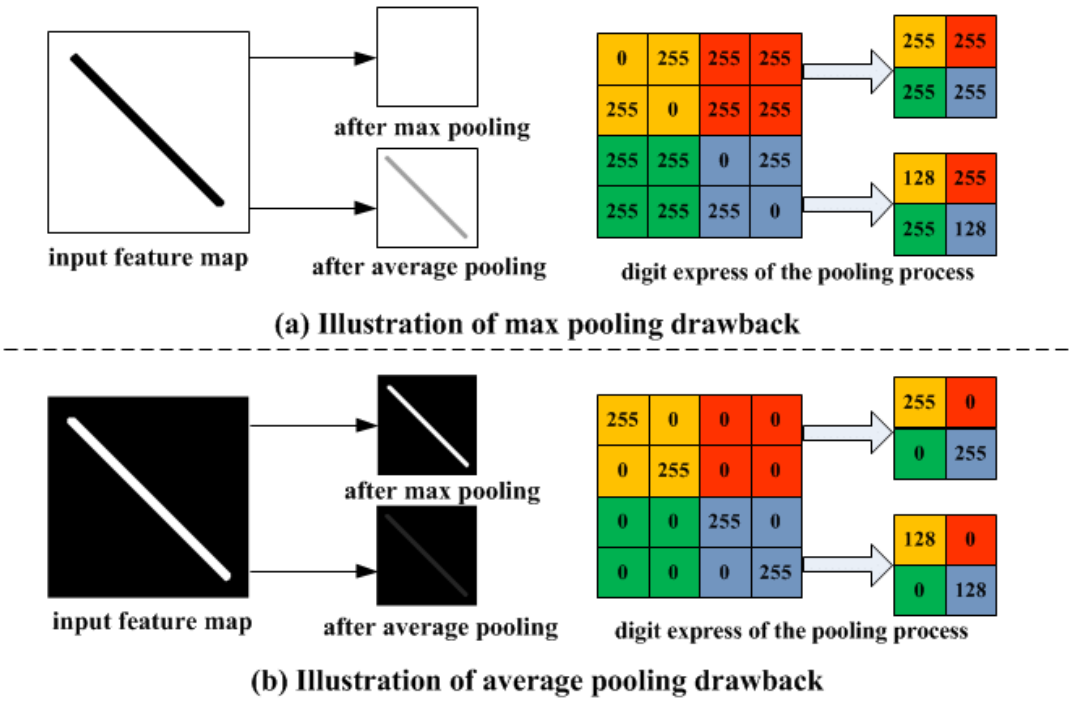

Figure 10 *The difference between using Max Pooling vs Average Pooling (Towards AI, 2022)*

Lastly, the FC or fully connected layers are usually found towards the end of CNN architecture. They are mainly used to optimise objectives such as class scores.

Gradient Boosting Model

The gradient boosting model was also used by Mandiant to detect obfuscated command line and has shown excellent results. To explain how this algorithm works, first, it needs to be explained what boosting is. Boosting is an ensemble technique that converts "weak learners" to a "strong learner". The strong learner is obtained from the additive model of weak learners. Weak learners are typically decision trees, and they have a high error rate. The algorithm first needs to have an initial weak learner in which all distributions have an equal weight assigned. The weak learner then makes predictions and for each wrongly classified distribution, the weights are increased while for each correctly classified distribution, the weights are decreased. Another weak learner is then made out of that, and the process repeats until the optimal results are obtained, i.e., errors are minimised, and predictions are correct. After each iteration, the algorithm should be one step closer to the final model.

In Gradient Boosting Model, a Loss function is used to estimate the model's ability to correctly make predictions with the given data. Depending on the problem, different Loss functions are used. For regression problems MSE or mean squared error is a good choice while for classification problems log-likelihood is commonly used. (Saini, 2021)

Random Forest Model

Random Forest is an algorithm made up of many individual decision trees. Each tree makes a prediction and the prediction with the most votes becomes the final prediction.

Like in Gradient Boosting, Random Forest combines multiple models but uses a bagging technique instead of boosting. Bagging chooses a random sample from the data set. Each model is trained independently, and they all make their predictions. All the results are then combined, and output is generated based on majority voting.

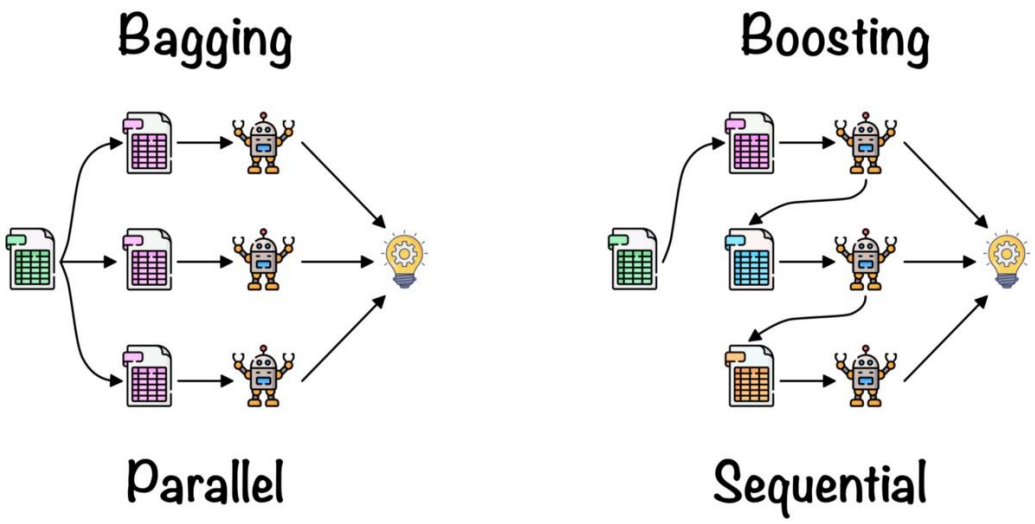

Figure 11 *Ensemble techniques: Bagging vs Boosting (López, 2021)*

Several models are usually more accurate than just one, making Radom Forest more successful than a single Decision Tree. However, for a Random Forest to be more successful, its trees must be diverse and acceptable.

Linear Regression Model

The Linear Regression model was used in the first program made for the detection of obfuscated PowerShell scripts, made by Daniel Bohannon and Lee Holme, and presented at the 2018 Black Hat conference. (Bohannon, 2018) Linear Regression is a model which assumes that a single output variable, *y*, can be calculated from a linear combination of the input variables, *x*. (Brownlee, 2020) This algorithm is based on supervised learning. Linear Regression can be Simple Linear Regression if it has only one independent variable or Multiple Linear Regression if it has multiple independent variables.

A Linear Regression model's primary goal is finding the best fit linear line and the optimal values of intercept and coefficients such that the error is minimised. Error is the difference between the actual value and predicted value and the goal is to reduce this difference. (Deepanshi, 2021)

Linear Regression is excellent in predicting output that is a continuous value, but it is not great to use for classification problems. To determine if CMD is obfuscated or not is a binary classification problem. That means the algorithm is looking at the probability of it being obfuscated. The probability range is between 1 and 0, where 1 may mean "obfuscated" and 0 - "not obfuscated". The problem is that Linear Regression predicts an absolute number, which can range outside 0 and 1. Results can be adjusted so that anything above 1 is 1 and anything below 0 is 0, but there are more suitable algorithms for classification. (Jing, 2019)

Existing machine learning solutions

As previously stated, researchers from Mandiant used machine learning to try and simplify the solution for Windows command line obfuscation. For their Featureless technique, they used Convolutional Neural Network and for their Feature-based technique, they chose Gradient Boosted Decision Tree. For the CNN model, they started by converting the raw text data into numeric form. For that, they used character-based encoding so that a real-valued number encoded each character type. For their Feature-based GBT model, they had to decide which features to include. Some of the features they decided to have were: the length of the command line, the number of carets in the command line, the count of pipe symbols, fractions of white space in the command line, the fraction of special characters, the entropy of the string as well as the frequency of the strings "cmd" and "power" in the command line. While none of the features alone is a clear indicator of obfuscation, they state that using them with flexible classifiers such as a Gradient Boosted Tree gives accurate results. (Hedge, 2018)

Their results showed that the CNN model had more misclassifications than the Gradient Boosted Tree model although the results were "nearly identical". They did not provide exact numbers, but they claim the F1-score, precision, and recall were close to 1.0 for GBT and slightly less accurate for CNN. (Hedge, 2018)

Adobe also decided to increase its detection ability by applying deep learning. They have open-sourced the results and the full code can be found on the Adobe GitHub page. The algorithm of choice for them was a Convolutional Neural network. Each command was represented by a series of characters and each character in the command was represented by a one-hot vector. In this vector, all the values are 0 except for the one index that represents which character it is in the string. They also included an extra bit whose function was to differentiate between uppercase and lowercase characters. The input to their model is a 74x4096 matrix because 73 characters were found most common in their data set and with the extra case bit, each character's one-hot vector is 74-dimensional. Each command is represented by its first 4096 characters. If the command is longer, the rest is discarded; if it is shorter, the command is padded with zeros. For the convolution layer, the algorithm takes several neighbouring characters, multiplies them by some weights, and outputs a one-dimensional vector which then has semantic meaning for those few characters. Applying more convolution layers increases its window size, i.e., the number of neighbouring characters. The bigger the window size, the more semantic meaning the convolutional layer can carry for each row. (Tang, 2021)

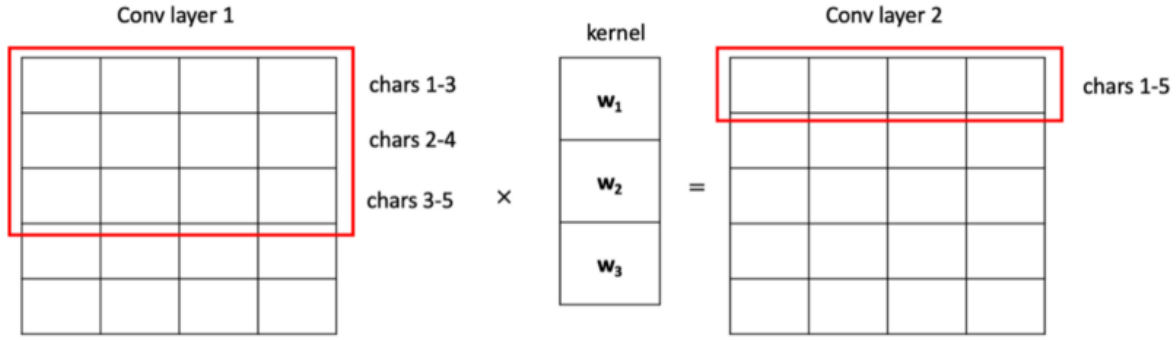

Figure 12 *How convolution layers are applied in Adobe's machine learning solution (Tang, 2021)*

The process is repeated to extract meaning from the command as a whole. After applying all convolution layers, a decision is made by taking an average of the CNN's output. This average is then run through a final fully connected layer to make the final output.

Their model achieved an F1 score of 0.9891. The model's errors were False Negatives, meaning the model classified obfuscated commands as not obfuscated. (Tang, 2021)

Limitations of existing machine learning approaches

To our knowledge, there have only been a few researchers who have significantly researched CMD obfuscation. Both existing approaches have been built based on the research presented in the DOSfuscation paper written by Daniel Bohannon. Mandiant's solution was built in 2018, just a few months after Bohannon presented his research. Although built in 2021, Adobe's solution was also based on Bohanon's research and they stated that their project was built upon previous work by FireEye (Mandiant's old corporate name) from 2017. (Tang, 2021)

The prime targets for CMD obfuscation are large companies and organisations which can have hundreds or even thousands of end devices where the obfuscation can be done. An effective detection tool should be able to automatically examine each command run in CMD on any device and send an alert to the security team if it detects obfuscation. Adobe's machine learning approach runs entirely in CMD and a single command or a set of commands must be entered manually into it to be examined. The program's classification result is then outputted in CMD in the form of 1s and 0s, where 1 indicates obfuscation and 0 non-obfuscation.

```
import obfuscation_detection as od

oc = od.ObfuscationClassifier(od.PlatformType.ALL)
commands = ['cmd.exe /c "echo Invoke-DOSfuscation"',
            'cm%windir:~ -4, -3%.e^Xe,;^,/^C",;,S^Et ^^o^=fus^cat^ion&,;,^se^T ^ ^ ^B^=o^ke-D^OS&&,;,s^Et^^
            'cat /etc/passwd']
classifications = oc(commands)

# 1 is obfuscated, 0 is non-obfuscated
print(classifications) # [0, 1, 0]
```

Figure 13 *Usage of Adobe's solution (Tang, 2021)*

Mandiant's machine learning approach is only described on the company's website. There is no software to download and test. Therefore, it cannot be evaluated or benchmarked against other solutions. That makes Adobe's solution the only solution available to us at present. As mentioned previously, Adobe tested and used only the CNN algorithm which was shown to be less accurate for both CMD and PowerShell obfuscation detection.

Technology Overview

In this section, various technologies will be researched to help create a new solution as a response to the limitations of the existing CMD obfuscation detection programs.

Programming Languages

Not a single programming language can be labelled as the best programming language for machine learning. Some languages are used more frequently than others, but the decision on what to use depends on the personal preference of the programmer as well as on the task the programmer is trying to accomplish. The top five most-used languages for machine learning, from most to least popular, according to a survey by Towards Data Science (Voskoglou, 2017), will be researched to decide which is the best one to use for this project.

Python

Python is a cross-platform, object-oriented language created in 1991. One of Python's strengths is that it comes with a complete standard library that allows the user to do all sorts of things with just one or a few lines of code. A library is a prewritten piece of code published by different sources, allowing users to reach some functionality or perform various actions to optimise the task. In addition to the standard library, Python has thousands of third-party libraries. (Summerfield, 2009)

Some of the most popular libraries in Python for machine learning are:

- TensorFlow
  - Open-source library for numerical computation and large-scale machine learning created by Google and available for Python, JavaScript, C++, and Java. (Yegulalp, 2022)
- Keras
  - Deep learning API which runs on top of TensorFlow. It was developed to enable fast experimentation. Prides on being simple, flexible and powerful. (Keras, n.d.)
- Pandas
  - An open-source project developed for high-level data structures and analysis. Allows merging, filtering, and gathering data from external sources. (Luashchuk, 2019)
- Scikit-learn
  - Offers simple and efficient tools for predictive data analysis. It can be used for machine learning algorithms such as clustering, regression, classification, linear and logistic regressions… (Pedregosa, et al., 2011)
- Caffe
  - Deep learning framework whose name stands for Convolutional Architecture for Fast Feature Embedding. Typically used for image classification and segmentation. (Berkeley AI Research, n.d.)

Another reason why Python is so popular is because of how easy to learn and simple it is. Because users do not need to worry about the code's complexity, they can focus entirely on the machine learning problem they are trying to solve.

However, Python does have its downsides. Python is an interpreted language which means it needs an interpreter instead of being compiled directly into native machine instructions before execution. That makes Python slower than compiled languages. Also, Python is not able to provide its users with low-level programming functionality as much as other programming languages. Furthermore, Python has limitations when it comes to multi-threading and parallel processing, which might be an issue if an algorithm that requires lots of computation time on large datasets is used. (Noema, 2022)

C++

C++ is an object-oriented programming language which began as an expanded version of C in 1979. (Schildt, 2003) C++ is efficient and it allows control of resources like CPU and memory management. With regard to machine learning, C++ is often used because of its speed. (Hashesh, 2020) It is probably the fastest language to use unless another language, e.g., Python, has a specific library which makes the task both simpler and faster. Listed below are several sophisticated C++ libraries that are helpful when it comes to developing a machine learning program. Previously mentioned libraries TensorFlow and Caffe, are also available to use in C++.

- Microsoft Cognitive Toolkit (CNTK)
  - Deep-learning toolkit which describes neural networks as a series of computational steps via a directed graph. Users can easily realize and combine model types such as feed-forward DNNs, convolutional networks (CNNs), and recurrent networks (RNNs/LSTMs). It can also be used with Python. (Microsoft, n.d.)
- Mlpack
  - Machine learning library which allows its users scalability, speed, and ease of use. It contains a number of machine learning algorithms, such as logistic regression, random forests, and k-means clustering. (Curtin, et al., 2018) It can also be used with Python.
- DyNet
  - Dynamic Neural Network Toolkit is designed to be efficient on either CPU or GPU. It is really good at natural language processing, graph structures, reinforcement learning etc. (Neubig, Michel, Xie, & Terenin, 2018) It can also be used with Python.

Java

Java is an object-oriented language created in 1996. Although it is similar to C and C++, it does not allow its users low-level programming functionalities like pointers. Regarding speed, Java is somewhere between Python and C++, not as fast as C++ but faster than Python. (GeeksforGeeks, 2022) Java Virtual Machine allows Java programs to run on any device and operating system and optimises program memory. Java's scalability makes it a great programming language for complex machine learning algorithms. Furthermore, several frameworks such as Spark, Kafka, Hadoop, Hive, Cassandra etc., are Java-based, run on the JVM, and provide functionalities for machine learning. (Hillier, 2021) Java also has several machine learning libraries such as Weka, a collection of machine

learning algorithms for data mining tasks; MOA which provides algorithms for classification, regression, clustering, and recommendations; Apache Mahout which can be used for classification, collaborative filtering, and clustering; Mallet, a specialised toolkit for natural language processing; and more.

R

R is a scripting language developed in 1993 and used for statistical data manipulation and analysis. (Matloff, 2011) It was inspired by the statistical language S developed by AT&T. The initial purpose of R was for statistical computing however nowadays it is commonly used for data science and machine learning as well. It is platform-independent, meaning it can easily run on Windows, Linux, and Mac. R has a wide variety of tools for machine learning. It can be used for classification, regression, time-series analysis, clustering, linear modelling, data cleansing, and R data wrangling. Another advantage of using R for machine learning is the availability of packages for data visualisation. R can be used to build both 2D and 3D graphic models.

A big downside to R is that it stores all objects in physical memory meaning it can be both slow and memory-consuming. (DAC.digital, n.d.) In one experiment, Python and R were used to execute the same function and Python code was 5.8 times faster than the R alternative. (Korstanje, 2020) Additionally, R is very different from other commonly used languages, so it can be challenging to learn.

JavaScript

JavaScript is a high-level, dynamic scripting language developed in 1995. It was made as a front-end language but the creation of Node.js gave it the ability to be used for backend development too. Recently the number of machine-learning JavaScript libraries has been growing. It supports TensorFlow.js which allows the creation of new machine learning models as well as the running and retention of existing ones in JavaScript. JavaScript also has a neural network library, Synaptic and image processing tools such as OpenCV.js. Math.js library has helped JavaScript become one of the languages used for machine learning because it provided it with much more mathematical flexibility and computing power. (Hillier, 2021)

Anaconda

Anaconda is an open-source distribution for Python and R and it is used to simplify package deployment and management. Anaconda comes with 250+ packages pre-installed and more than 7500 additional open-source packages can be installed. Conda is Anaconda's package management system which is used to download, run and update the packages in just a few clicks.

Along with the pre-installed popular data science packages, Anaconda comes with a desktop GUI, making it easier for beginners to work on their first data science project. Another significant benefit of using Anaconda is the security aspect. It can identify security vulnerabilities (CVEs) a package and its dependencies are exposed to in order to block unsafe packages. (Anaconda Inc.)

Sysmon

System Monitor or Sysmon is a Windows system service and device driver. Its job is to monitor and log system activity to the Windows event log. There are 23 types of events Sysmon monitors on Windows. It is widely used for incident response and threat hunting, providing detailed information about process creation, network connections, and file modifications. (Russinovich & Garnier, 2022) When an event is logged, Sysmon provides various information regarding it, as seen in Figure 14.

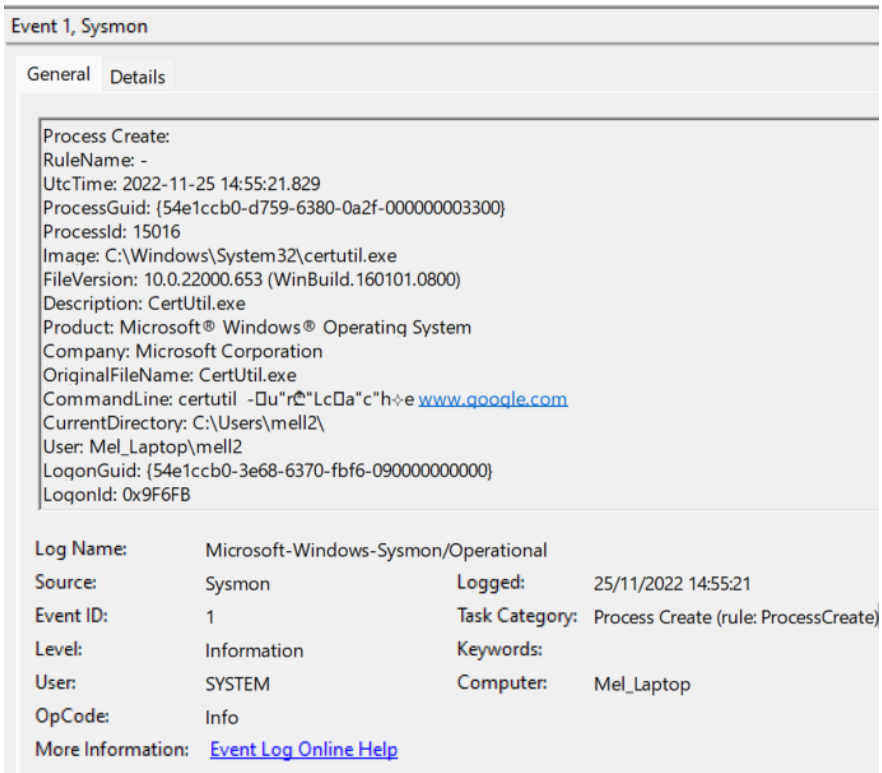

Figure 14 *Sysmon event example.*

Conclusion

CMD obfuscation is a growing concern in cyber security. It allows malicious actors to execute malicious activities without being detected by disguising otherwise obvious commands with an obfuscated version of them. There are only two approaches to solving the problem, machine learning and traditional-style detections.

Traditional-style detections, like Sigma rules or regex, work perfectly on unobfuscated commands but are not a suitable solution when it comes to obfuscation. Creating a rule which is able to cover all the obfuscation methods would be difficult and hard to maintain. A rule of that complexity would take such a long time to process each of the commands that it would prove to be useless. That means relying on traditional-style detections when it comes to process execution events brings the entire organization at risk. On the other hand, machine learning's ability to learn based on data and experience makes it an excellent solution for detecting anomalies. High precision and recall scores of the two existing CMD obfuscation detection tools prove that machine learning is the best approach. Even though two different machine learning solutions for CMD obfuscation have already been developed, their limitations point that a new, improved tool is needed if we want to implement it for real-life obfuscation detection.

There are numerous machine learning algorithms which can be used for a classification problem such as this one. Out of all of them, four showed promising results in either CMD or PowerShell obfuscation detection. This research showed that three of those, Convolutional Neural Network, Gradient Boosting and Random Forest are promising enough to be worth testing with the project's datasets. The Gradient Boosting algorithm, which uses a boosting technique to convert weak models into a strong one, has shown the most accurate results whenever it was used for obfuscation detection.

Multiple languages were researched for this project, and Python proved to be the most suitable one. Despite not being the fastest, its ease of use and the number of libraries available for machine learning put it in the first spot. For this project, libraries TensorFlow and Keras will be used along with the Anaconda distribution. TensorFlow is great for building machine learning models for experts and beginners, while Keras, as its official high-level API, should be valuable for building the CNN model. Using Anaconda will make it easier to install both of those libraries and any other that may be used in future.

Lastly, this project aims to incorporate Sysmon to automate obfuscation detection. As stated previously, Sysmon logs process creation with the full command line of the current process and the parent process. This project will try to extract every command line from Sysmon event logs and automatically process each to see if obfuscation occurred.

References

Anaconda Inc. (n.d.). *Anaconda The World's Most Popular Data Science Platform.* Retrieved November 20, 2022, from Anaconda: https://cdn2.assets-servd.host/voracious-blesbok/production/Resources/assets/ANA_Why-Anaconda-Guide-3.pdf

Avira Protection Labs. (2020, April 01). *CertUtil abused by attackers to spread threats*. Retrieved October 23, 2022, from Avira: https://www.avira.com/en/blog/certutil-abused-by-attackers-to-spread-threats

Berkeley AI Research. (n.d.). *Caffe*. Retrieved November 19, 2022, from Berkeley Vision: https://caffe.berkeleyvision.org/

Beukema, W. (n.d.). *Windows Command-Line Obfuscation*. Retrieved October 23, 2022, from WietzeBeukema: https://www.wietzebeukema.nl/blog/windows-command-line-obfuscation

Bohannon, D. (2018, March 22). *DOSfuscation: Exploring the Depths of Cmd.exe Obfuscation and Detection Techniques*. Retrieved November 06, 2022, from Mandiant: https://www.mandiant.com/resources/blog/dosfuscation-exploring-obfuscation-and-detection-techniques

Brownlee, J. (2020, August 15). *Linear Regression for Machine Learning*. Retrieved November 05, 2022, from Machine Learning Mastery: https://machinelearningmastery.com/linear-regression-for-machine-learning/

Burns, E. (2021, March). *Machine Learning*. Retrieved November 04, 2022, from TechTarget: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

Curtin, R. R., Edel, M., Lozhnikov, M., Mentekidis, Y., Ghaisas, S., & Zhang, S. (2018, June 18). *mlpack 3: a fast, flexible machine learning library.* Retrieved November 19, 2022, from mlpack: https://www.mlpack.org/static/pub/2018mlpack.pdf

DAC.digital. (n.d.). *Which is better for AI - Python or R?* Retrieved November 18, 2022, from DAC.digital: https://dac.digital/which-is-better-for-ai-python-or-r/

Deepanshi. (2021, May 25). *All you need to know about your first Machine Learning model – Linear Regression*. Retrieved November 06, 2022, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/

GeeksforGeeks. (2022, November 15). *Java*. Retrieved November 19, 2022, from GeeksforGeeks: https://www.geeksforgeeks.org/java/

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep Learning.* Cambridge, MA: MIT Press.

Hashesh, A. (2020, May 30). *When Should You Learn Machine Learning using C++?* Retrieved November 18, 2022, from Medium: https://medium.com/ml2b/when-should-you-learn-machine-learning-using-c-6edd719f95ff

Hedge, V. (2018, November 29). *Obfuscated Command Line Detection Using Machine Learning*. Retrieved November 06, 2022, from Mandiant:

https://www.mandiant.com/resources/blog/obfuscated-command-line-detection-using-machine-learning

Hillier, W. (2021, December 27). *What's the Best Language for Machine Learning?* Retrieved November 18, 2022, from Career Foundry: https://careerfoundry.com/en/blog/data-analytics/best-machine-learning-languages/

IBM. (2022). *How much does a data breach cost in 2022?* Retrieved November 04, 2022, from IBM: https://www.ibm.com/uk-en/security/data-breach

Jing, H. (2019, May 07). *Why Linear Regression is not suitable for Classification*. Retrieved November 06, 2022, from Towards Data Science: https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

Keras. (n.d.). *About Keras*. Retrieved November 20, 2022, from Keras: https://keras.io/about/

Korstanje, J. (2020, May 22). *Is Python faster than R?* Retrieved November 18, 2022, from Towars Data Science: https://towardsdatascience.com/is-python-faster-than-r-db06c5be5ce8

Kurama, V. (2019). *Gradient Boosting In Classification: Not a Black Box Anymore!* Retrieved November 05, 2022, from PaperspaceBlog: https://blog.paperspace.com/gradient-boosting-for-classification/

López, F. (2021, January 11). *Ensemble Learning: Bagging & Boosting*. Retrieved from Towards Data Science: https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

Luashchuk, A. (2019, May 29). *Why I Think Python is Perfect for Machine Learning and Artificial Intelligence*. Retrieved November 20, 2022, from Towards Data Science: https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6

Mandal, M. (2021, May 01). *Introduction to Convolutional Neural Networks (CNN)*. Retrieved November 05, 2022, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Matloff, N. (2011). *The art of R programming* (9th ed.). San Francisco: William Pollock.

MDN. (2022, September 14). *Regular expressions*. Retrieved October 23, 2022, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Microsoft. (n.d.). *CNTK*. Retrieved November 19, 2022, from Github: https://github.com/microsoft/CNTK

Mitchell, T. (1997). *Machine learning* (1st ed.). New York: McGraw-Hill.

Moe, O., Bayne, J., Richard, C., Spehn , C., Liam , & Wietze . (n.d.). *Living Off The Land Binaries, Scripts and Libraries*. Retrieved October 23, 2022, from Lolbas Project: https://lolbas-project.github.io/

Neubig, G., Michel, P., Xie, Q., & Terenin, A. (2018, March 23). *DyNet documentation.* Retrieved November 19, 2022, from Github: https://github.com/clab/dynet/blob/master/doc/source/index.rst

Nicholson, C. (n.d.). *Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined*. Retrieved November 13, 2022, from A.I. Wiki: https://wiki.pathmind.com/accuracy-precision-recall-f1

Noema, Y. (2022, February). *What are the Pros and Cons of Using Python for Machine Learning?* Retrieved November 20, 2022, from Tealfeed: https://tealfeed.com/pros-cons-using-python-machine-learning-5ssdx

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Russinovich, M., & Garnier, T. (2022, November 10). *Sysmon v14.12*. Retrieved November 25, 2022, from Microsoft: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

Saini, A. (2021, September 20). *Gradient Boosting Algorithm: A Complete Guide for Beginners*. Retrieved November 06, 2022, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

Schildt, H. (2003). *C++: The Complete Reference, Fourth Edition.* Berkeley: Brandon A. Nordin.

Sharma, A. (2022, June 16). *Sigma rules explained: When and how to use them to log events*. Retrieved October 23, 2022, from CSO: https://www.csoonline.com/article/3663691/sigma-rules-explained-when-and-how-to-use-them-to-log-events.html

Summerfield, M. (2009). *Programming in Python 3: a complete introduction to the Python language* (1st ed.). Boston: Pearson Education.

Tang, W. (2021, August 24). *Using Deep Learning to Better Detect Command Obfuscation*. Retrieved November 06, 2022, from Adobe Tech Blog: Using Deep Learning to Better Detect Command Obfuscation

Towards AI. (2022, August 16). *Introduction To Pooling Layers In CNN*. Retrieved November 06, 2022, from Towards AI: https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn

Tunggal, A. T. (2022, October 03). *What is the Cost of a Data Breach in 2022?* Retrieved November 13, 2022, from UpGuard: https://www.upguard.com/blog/cost-of-data-breach

Voskoglou, C. (2017, May 5). *What is the best programming language for Machine Learning?* Retrieved November 18, 2022, from Towards Data Science: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7

Yegulalp, S. (2022, June 03). *What is TensorFlow? The machine learning library explained*. Retrieved November 19, 2022, from InfoWorld: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html