##### Malware Detection Using Artificial Intelligence

##### A PROJECT REPORT

###### **Submitted by**

##### NAME OF THE CANDIDATE(S)

***for the award of the degree***

***of***

##### NAME OF THE DEGREE

*IN*

BRANCH OF STUDY

****

*INTEGRAL UNIVERSITY LUCKNOW*

MONTH & YEAR

*INTEGRAL UNIVERSITY: LUCKNOW*

**UNDERTAKING**

We, <Name of students>, student of B.Tech. (CSE/IT), hereby declare that the work detailed in this Project entitled <“Topic Name”> submitted to the Department of Computer Science and Engineering, Integral University, Lucknow for the award of the Bachelor of Technology degree is our original work. We have neither plagiarized nor submitted this work for the award of any other degree. In case, this undertaking is found incorrect, We accept that my degree may be unconditionally withdrawn.

Date:

Venue: Lucknow

<Name of Student 1> < Signature of Student 1>

<Name of Student 2> < Signature of Student 2>

<Name of Student 3> < Signature of Student 3>

<Name of Student 4> < Signature of Student 4>

(A typical specimen of Certificate)

<Font Style Times New Roman>

*INTEGRAL UNIVERSITY: LUCKNOW*

<Font Style Times New Roman – size -18>

**CERTIFICATE**

This is to certify that the work contained in this project report entitled <”Topic Name”> by

<Student Name1>, <Student Name2> , <Student Name3> , <Student Name4> is a faithful record of work that has been carried out by the students, under my supervision and the level of work is good for submission. To the best of my/our knowledge, this work has not been submitted for award of any degree or diploma to this University or elsewhere.

<<Signature of the Guide>>

<<Name of the Guide >>

<<Academic Designation>>

<<Department>>

<<Full address of the Dept & College >>

*INTEGRAL UNIVERSITY: LUCKNOW*

**RECOMMENDATION**

On the basis of the declaration submitted by <Student Names>, internal assessment carried out by department on date dd/mm/yyyy and the certificate issued by the Guide <Guide Name> & <Co Guide> the work entitle “<Topic Name> submitted to department of CSE, is recommended for final examination.

Signature B.Tech Project Coordinator

Name & Designation Signature of HOD

Name & Designation

Date:\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_

TABLE OF CONTENTS

1. Cover Page & Title Page

2.Declaration by the group members

3. Certificate (Certificate from Guide )

4. Recommendation (From B.Tech Project Coordinator & HOD)

5. Certificate by the Company (If Live project)

6. Acknowledgement

7. Abstract

8. Table of content

9. List of Figure

10. List of Table

11. List of Symbols, Abbreviations and Nomenclature

. 12. Chapters

12.1 Introduction

12.2 Problem Identification & Feasibility Study

12.3 Requirement Analysis

12.4 Review of Previous work

12.5 Proposed Work (Framework or Algorithm)

12.6 Hardware & Software Specification

12.7 Design

Context Diagram

Data Flow Diagrams

Entity Relationship Diagrams

Flowchart

Snapshot of the Project

12.8. Result & Snapshot

12.9 Conclusion

13. References

14. Bio data of each group member

**Abstract**

Machine Learning is a subfield of computer science that aims to give computers the ability to learn from data instead of being explicitly programmed, thus leveraging the petabytes of data that exists on the internet nowadays to make decisions, and do tasks that are somewhere impossible or just complicated and time consuming for us humans.

Malware is one the imminent threats that companies and users face every day. Whether it is a phishing email or an exploit delivered throughout the browser, coupled with multiple evasion methods and other security vulnerabilities, it is a proven fact that nowadays defense systems cannot compete. The availability of frameworks such as Veil, Shelter, and others are known to be used by professionals when conducting pentesting work and are known to be quite effective.

We will use Machine Learning to detect Malware without having to use neither a signature detection nor a behavioral analysis.

Traditional security product uses virus scanner to detect malicious code, these scanner uses signature which created by reverse engineering a malware. But with malware that became polymorphic or metamorphic the traditional signature based detection method used by anti-virus is no long effective against the current issue of malware (Willems, G., Holz, T. & Freiling, F., 2007). In current anti-malware products, there are two main task to be carried out from the malware analysis process, which are malware detection and malware classification. In this paper, I am focusing on malware detection. The main objective of malware detection is to be able to detect malware in the system. There are two type of analysis for malware detection which are dynamic analysis and static analysis. For effective and efficient detection, the uses of feature extraction are recommended for malware detection (Ahmadi, M. et al., 2016). There are various type of detection method, the method that we are using will be detecting through hex and assembly file of the malware. Feature will be extracted from both hex view and assembly view of malware files. After extracting feature to its category, all category is to be combine into one feature vector for the classifier to run on them (Ahmadi, M. et al., 2016). For feature selection, separating binary file into blocks to be compare the similarities of malware binaries. This will reduce the analysis overhead which cause the process to be faster (Kim, T.G., Kang, B. & Im, E.G., 2013). To build a learning algorithm, feature that are extracted with the label will be undergo classification with using any classification method for example Random Forest, Neural Network, N-gram, KNN and many others, but Support Vector Machine (VCM) is recommended for the presence of noise in the extracted feature and the label (Stewin, P. & Bystrov, I., 2016). As to generate result, the learning model is to test with dataset with label to generate a graph which indicate detection rate and false positive rate. To find the best result, repeat the process using many other classification and create learning model to test on the same dataset. The best result will the one graph that has the highest detection rate and lowest false positive rates (Lanzi, A. et al., 2010).

**List of Figures**

|  |  |
| --- | --- |
| **Page No.** | **Description of image** |
| 24 | Theoretical / Conceptual Framework |
| 26 | The general workflow of the machine learning process |

**Introduction**

**Malware**

Malware, or malicious software, is any program or file that is harmful to a computer user. Types of malware can include computer viruses, worms, Trojan horses and spyware. These malicious programs can perform a variety of different functions such as stealing, encrypting or deleting sensitive data, altering or hijacking core computing functions and monitoring users' computer activity without their permission.

Malware authors use a variety of physical and virtual means to spread malware that infect devices and networks. For example, malicious programs can be delivered to a system with a USB drive or can spread over the internet through drive-by downloads, which automatically download malicious programs to systems without the user's approval or knowledge. Phishing attacks are another common type of malware delivery where emails disguised as legitimate messages contain malicious links or attachments that can deliver the malware executable to unsuspecting users. Sophisticated malware attacks often feature the use of a command-and-control server that allows threat actors to communicate with the infected systems, exfiltrate sensitive data and even remotely control the compromised device or server.

Emerging strains of malware include new evasion and obfuscation techniques that are designed to not only fool users but security administrators and anti-malware products as well. Some of these evasion techniques rely on simple tactics, such as using web proxies to hide malicious traffic or source IP addresses. More sophisticated threats include polymorphic malware, which can repeatedly change its underlying code to avoid detection from signature-based detection tools, anti-sandbox techniques, which allow the malware to detect when it is being analyzed and delay execution until after it leaves the sandbox, and fileless malware, which resides only in the system's RAM in order to avoid being discovered.

**Common types of malware**

Different types of malware contain unique traits and characteristics. Types of malware include:

* A **virus** is the most common type of malware which can execute itself and spread by infecting other programs or files.
* A  **worm**  can self-replicate without a host program and typically spreads without any human interaction or directives from the malware authors.
* **A Trojan horse**   is  designed to appear as a legitimate program in order to gain access to a system. Once activated following installation, Trojans can execute their malicious functions.
* **Spyware**   is  made to collect information and data on the device user and observe their activity without their knowledge.
* [**Ransomware**](https://searchsecurity.techtarget.com/definition/ransomware) is designed to infect a user's system and encrypt the data. Cybercriminals then demand a ransom payment from the victim in exchange for decrypting the system's data.
* A [**rootkit**](https://searchsecurity.techtarget.com/definition/rootkit)  is created  to obtain administrator-level access to the victim's system. Once installed, the program gives threat actors root or privileged access to the system.
* A **backdoor** virus or remote access Trojan ([**RAT**](https://searchsecurity.techtarget.com/definition/RAT-remote-access-Trojan)) secretly creates a backdoor into an infected system that allows threat actors to remotely access it without alerting the user or the system's security programs.
* [**Adware**](https://searchsecurity.techtarget.com/definition/adware) is used to track a user’s browser and download history with the intent to display pop-up or banner advertisements that lure the user into making a purchase. For example, an advertiser might use [**cookies**](https://searchsoftwarequality.techtarget.com/definition/cookie) to track the web pages a user visits to better target advertising.
* [**Keyloggers**](https://searchsecurity.techtarget.com/definition/keylogger), also called system monitors, are used to see nearly everything a user does on their computer. This includes emails, opened web-pages, programs and keystrokes.

**Mobile malware**

Malware can also be found on mobile phones and can provide access to the device's components such as the camera, microphone, GPS or accelerometer. Malware can be contracted on a mobile device if the user downloads an unofficial application or if they click on a malicious link from an email or text message. A mobile device can also be infected through a Bluetooth or Wi-Fi connection.

Malware is found much more commonly on devices that run the Android OS comparatively to iOS devices. Malware on Android devices is usually downloaded through applications. Signs that an Android device is infected with malware include unusual increases in data usage, a quickly dissipating battery charge or calls, texts and emails being sent to the device contacts without the user's knowledge. Similarly, if a user receives a message from a recognized contact that seems suspicious, it may be from a type of a mobile malware that spreads between devices.

Apple iOS devices are rarely infected with malware because Apple carefully vets the applications sold in the App Store. However,  it is still possible for an iOS device to be infected by opening an unknown link found in an email or text message. iOS devices will become more vulnerable if [jailbroken](https://whatis.techtarget.com/definition/jailbreaking).

**Malware Detection methods**

All malware detection techniques can be divided into signature-based and behavior-based methods. Before going into these methods, it is essential to understand the basics of two malware analysis approaches: static and dynamic malware analysis. As it implies from the name, static analysis is performed “statically”, i.e. without execution of the file. In contrast, dynamic analysis is conducted on the file while it is being executed for example in the virtual machine.

Static analysis can be viewed as “reading” the source code of the malware and trying to infer the behavioral properties of the file. Static analysis can include various techniques (Prasad, Annangi and Pendyala 2016) :

1. **File Format Inspection**: file metadata can provide useful information. For example, Windows PE (portable executable) files can provide much information on compile time, imported and exported functions, etc.

2. **String Extraction**: this refers to the examination of the software output (e.g. status or error messages) and inferring information about the malware operation.

3. **Fingerprinting**: this includes cryptographic hash computation, finding the environmental artifacts, such as hardcoded username, filename, registry strings.

4. **AV scanning**: if the inspected file is a well-known malware, most likely all anti-virus scanners will be able to detect it. Although it might seem irrelevant, this way of detection is often used by AV vendors or sandboxes to “confirm” their results.

5. **Disassembly**: this refers to reversing the machine code to assembly language and inferring the software logic and intentions. This is the most common and reliable method of static analysis.

Static analysis often relies on certain tools. Beyond the simple analysis, they can provide information on protection techniques used by malware. The main advantage of static analysis is the ability to discover all possible behavioral scenarios. Researching the code itself allows the researcher to see all ways of malware execution, that are not limited to the current situation. Moreover, this kind of analysis is safer than dynamic, since the file is not executed and it cannot result in bad consequences for the system. On the other hand, static analysis is much more time-consuming. Because of these reasons it is not usually used in real-world dynamic environments, such as anti-virus systems, but is often used for research purposes, e.g. when developing signatures for zero-day malware.

Another analysis type is dynamic analysis. Unlike static analysis, here the behavior of the file is monitored while it is executing and the properties and intentions of the file are inferred from that information. Usually, the file is run in the virtual environment, for example in the sandbox. During this kind of analysis, it is possible to find all behavioral attributes, such as opened files, created mutexes, etc. Moreover, it is much faster than static analysis. On the other hand, the static analysis only shows the behavioral scenario relevant to the current system properties. For example, if our virtual machine has Windows 7 installed, the results might be different from the malware running under Windows 8.1.

Now, having the background on malware analysis, we can define the detection methods. The signature-based analysis is a static method that relies on predefined signatures. These can be file fingerprints, e.g. MD5 or SHA1 hashes, static strings, file metadata. The scenario of detection, in this case, would be as follows: when a file arrives at the system, it is statically analyzed by the antivirus software. If any of the signatures is matched, an alert is triggered, stating that this file is suspicious. Very often this kind of analysis is enough since wellknown malware samples can often be detected based on hash values.

However, attackers started to develop malware in a way that it can change its signature. This malware feature is referred to as polymorphism. Obviously, such malware cannot be detected using purely signature-based detection techniques. Moreover, new malware types cannot be detected using signatures, until the signatures are created. Therefore, AV vendors had to come up with another way of detection – behavior-based also referred to as heuristicsbased analysis. In this method, the actual behavior of malware is observed during its execution, looking for the signs of malicious behavior: modifying host files, registry keys, establishing suspicious connections. By itself, each of these actions cannot be a reasonable sign of malware, but their combination can raise the level of suspiciousness of the file. There is some threshold level of suspiciousness defined, and any malware exceeding this level raises an alert. (Harley and Lee 2009). The accuracy level of heuristics-based detection highly depends on the implementation. The best ones utilize the virtual environment, e.g. the sandbox to run the file and monitor its behavior. Although this method is more timeconsuming, it is much safer, since the file is checked before actually executing. The main advantage of behavior-based detection method is that in theory, it can identify not only known malware families but also zero-day attacks and polymorphic viruses. However, in practice, taking into account the high spreading rate of malware, such analysis cannot be considered effective against new or polymorphic malware.

**Need for machine learning**

As stated before, malware detectors that are based on signatures can perform well on previously-known malware, that was already discovered by some antivirus vendors. However, it is unable to detect polymorphic malware, that has an ability to change its signatures, as well as new malware, for which signatures have not been created yet. In turn, the accuracy of heuristics-based detectors is not always sufficient for adequate detection, resulting in a lot of false-positives and false-negatives. (Baskaran and Ralescu 2016). Need for the new detection methods is dictated by the high spreading rate of polymorphic viruses. One of the solutions to this problem is reliance on the 11 heuristics-based analysis in combination with machine learning methods that offer a higher efficiency during detection. When relying on heuristics-based approach, there has to be a certain threshold for malware triggers, defining the amount of heuristics needed for the software to be called malicious. For example, we can define a set of suspicious features, such as “registry key changed”, “connection established”, “permission changed”, etc. Then we can state, that any software, that triggers at least five features from that set can be called malicious. Although this approach provides some level of effectiveness, it is not always accurate, since some features can have more “weight” than others, for example, “permission changed” usually results in more severe impact to the system than “registry key changed”. In addition to that, some feature combinations might be more suspicious than features by themselves.

To take these correlations into account and provide more accurate detection, machine learning methods can be used.

**Machine Learning**

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

**Some machine learning methods**

Machine learning algorithms are often categorized as supervised or unsupervised.

* **Supervised machine learning algorithms**can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
* In contrast, **unsupervised machine learning algorithms**are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
* **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiringunlabeled data generally doesn’t require additional resources.
* **Reinforcement machine learning algorithms**is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

**Problem Identification**

As exposed in an article published a short while ago, on the **6 need-to-know attributes of advanced cyber attacks**, one of the main ways in which second generation malware is challenging the security industry is **its capacity to evade detection**.

**Malware authors are not only trying to outdo themselves**, but also keep one step ahead of the cyber security industry. What makes them successful is that they have the resources (especially the time) to test version after version, and to incrementally enhance their tactics up to the point where malware can infect a system and **go undetected for months**.

There’s no reason to sugarcoat this: cyber security specialists are struggling and sometimes it makes it complicated for users to build their cyber defenses, because of the multiple fronts they need to cover. But the experts are aware of the challenges and they’re working to come up with new, more effective methods for cyber protection. This is especially important since **there is no such thing as a panacea** when protecting online assets.

Virtually every week a new report surfaces about a large, blue chip company with deep financial resources that has been breached. These companies typically invest in and deploy state-of-the-art security tools, yet attackers are still able to penetrate their lines of defense. To make matters worse, many attacks often go undetected for months. Let’s examine how this can happen.

**Attack vectors**

Every breach must exploit at least one attack vector in order to install persistent malware on the organization's network. Advanced attackers often use multi-stage malware, which would initially only install a small backdoor. This enables more complex tools to be deployed on the machine and network later on.

The primary malware installation, sometimes referred as an infection, can be achieved using several attack vectors. The goal is always to run malicious code. Some of the most common attack vectors are:

1. **Browser-based social engineering:** where a user is tricked into clicking on a legitimate-looking URL which in turn triggers code execution using browser or browser-plugin vulnerabilities in Java and Flash. More advanced attacks can hide in legitimate traffic without requiring any user-interaction. These are commonly referred to as drive-by downloads.
2. **Email-based social engineering and spear phishing:** where a user receives an email that contains a hidden or visible binary, which executes when the user clicks on it.
3. **Credential theft:** when guessed or stolen credentials are used to access a remote machine and execute (malicious) code, such as installing a backdoor.

**Evasion techniques**

To evade detection, during and after installation, malware uses five primary techniques.

1. **Wrapping.** This process attaches the malicious payload (the installer or the malware itself) to a legitimate file. When the legitimate file is installed, so is the malicious payload (which usually installs before the legitimate file does). Using static signatures to detect wrapper files is largely ineffective since new ones are easily and regularly created and often generates false positives. This technique is commonly used by Windows and OS X malware distributed via pirated software and P2P networks. IceFog is a well-known malware commonly wrapped with a legitimate-looking CleanMyMac application and used to target OS X users. On the Windows platform, OnionDuke has been used with legitimate Adobe installers shared over Tor networks to infect machines.
2. **Obfuscation**. This involves modifying high level or binary code it in a way that does not affect its functionality, but completely changes its binary signature. Obfuscation was originally used to protect legitimate software against reverse-engineering and piracy. Malware authors have adopted the technique to bypass antivirus engines and impair manual security research. Using XOR encoding is one way to do this. Hiding process and file names, registry entries, URLs and other useful information can significantly slow down the investigation/reverse engineering of new malware samples.
3. **Packers.** These software tools are used to compress and encode binary files, which is another form of obfuscation. At runtime, the packer, which is typically embedded with the malicious binary, will "unpack" the payload into memory and execute it. There are a handful of common packing mechanisms in use today such as UPX, PECompact, Armadillo and others. These techniques are extremely effective at circumventing static signature engines.
4. **Anti-debugging.** Like obfuscation, anti-bugging was originally created by software developers to protect commercial code from reverse-engineering. Anti-debugging can prevent a binary from being analyzed in an emulated environments such as virtual machines, security sandbox, and others. For example, the ZeroAccess malware implemented a self-debugging technique in order to block external debugging attempts. Another example is malware attempting to delay its execution (or sleep) for an extended period of time. This is useful for bypassing sandboxing solutions since these only keep binaries in an emulated environment for a specific period of time before classifying them as benign and releasing them to the network.
5. **Targeting.** This technique is implemented when malware is designed to attack a specific type of system (e.g. Windows XP SP 3), application (e.g. Internet Explorer 10) and/or configuration (e.g. detecting a machine not running VMWare tools, which is often a telltale sign for usage of virtualization). Targeting ensures that the malware is only triggered and installed when specific conditions are met, which enables it to evade detection in sandboxes because they do not resemble the host being attacked.

Just as malware's evasion techniques continue to evolve, so must our security measures. There is much work being done in the industry to move beyond traditional static signature-based security to behavior-based profiling, analytics and real-time information sharing between security solutions. One thing we have learned from researching the malware techniques described above is the closer we can place security to the targeted asset, the more likely we will be able to detect and stop it.

**Feasibilty Study**

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

**TECHNICAL FEASIBILITY**

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

In the creation of this project we did not faced any type of technical problem. We had all the tools required for the maintenance of the software and obviously a good internet connection was required to run the project without any obstacles.

**ECONOMIC FEASIBILITY**

Economic feasibility analysis is the most commonly used method for determining the efficiency of a new project. It is also known as cost analysis. It helps in identifying profit against investment expected from a project. Cost and time are the most essential factors involved in this field of study.

The development of this project do not cost any high amount of money and is economically feasible as it only requires Anaconda platform so that the tools can be installed and a good internet connection so that the details of the books can fetched properly.

**LEGAL FEASIBILTY**

The development of this project is legally feasible because the websites like AMAZON and SNAPDEAL allows data scraping from their websites and we have not done any illegal process while the development of the project.

**OPERATIONAL FEASIBILTY**

The project requires the Anaconda5 platform, the configuration settings, auth keys and a good internet connection for the smooth working of the project. The auth keys provides the access to the websites and a good internet connection will help us to fetch the details from different websites on a particular interface.

**Review of Previous Work**

The most common protection against malware is antivirus (AV) software. Despite what the name anti-virus suggests, anti-virus can also detect and possibly remove categories of malware besides viruses. A typical AV system works by scanning files during load time for known signatures, typically code strings, of malware. Figure 1 shows how anti-virus signatures are prepared: Honeypots collect malware and non-malware which are then analyzed by humans to create signatures. These signatures are then delivered to the host anti-virus software periodically. A complementary approach to signature-based detection is also used in practice [7]. In reputation based AV detection, users anonymously send cryptographic signatures of executables to the AV vendor. The AV vendor then determines how often an executable occurs in a large population of its users to predict if an executable is malware: often, uncommon executable signatures occurring in small numbers are tagged as malware. This system is reported to be effective against polymorphic and metamorphic viruses but does not work against non-executable threats such as malicious pdfs and doc files [8]. Further it requires users to reveal programs installed on their machine to the AV vendor and trust the AV vendor not to share this secret.

So, as we can see that Artificial Intelligence has not been applied in the field of malware detection which cannot be detected by any traditional AV, there is no review of past work.

**Proposed Work**

**Theoretical / Conceptual Framework**

Data Set

Evaluation

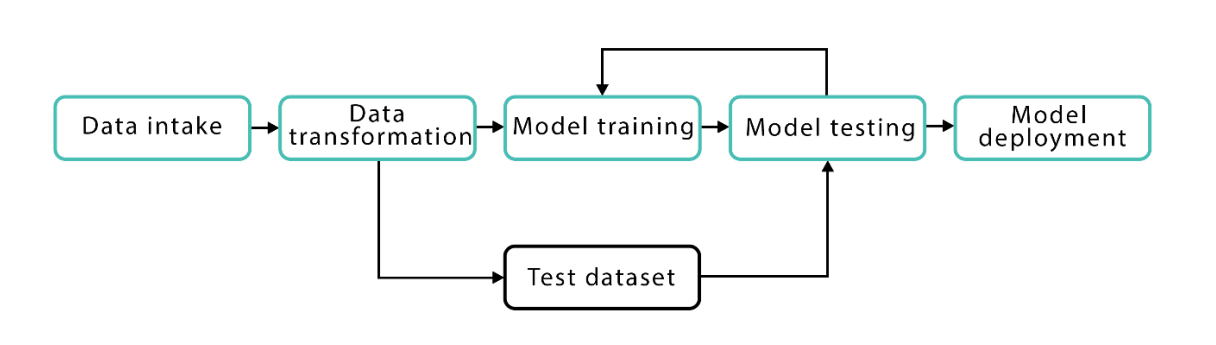
Feature Extraction

Classification

Although not widely implemented, the concept of machine learning methods for malware detection is not new. Several types of studies were carried out in this field, aiming to figure the accuracy of different methods. In his paper “Malware Detection Using Machine Learning” Dragos Gavrilut aimed for developing a detection system based on several modified perceptron algorithms. For different algorithms, he achieved the accuracy of 69.90%- 96.18%. It should be stated that the algorithms that resulted in best accuracy also produced the highest number of false-positives: the most accurate one resulted in 48 false positives. The most ”balanced”s algorithm with appropriate accuracy and the low false-positive rate had the accuracy of 93.01%.

MACHINE LEARNING METHODS This chapter gives a theoretical background on machine learning methods, needed for understanding the practical implementation. First, the overview of the machine learning field is discussed, followed by the description of methods relevant to this study. These methods include k-Nearest Neighbors, Decision Trees, Random Forests, Support Vector Machines and Naive Bayes. 3.1 Machine Learning Basics The rapid development of data mining techniques and methods resulted in Machine Learning forming a separate field of Computer Science. It can be viewed as a subclass of the Artificial Intelligence field, where the main idea is the ability of a system (computer program, algorithm, etc.) to learn from its own actions. It was firstly referred to as "field of study that gives computers the ability to learn without being explicitly programmed" by Arthur Samuel in 1959. A more formal definition is given by T. Mitchell: "A computer program is said to learn 13 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." (Mitchell 1997).

The basic idea of any machine learning task is to train the model, based on some algorithm, to perform a certain task: classification, clusterization, regression, etc. Training is done based on the input dataset, and the model that is built is subsequently used to make predictions. The output of such model depends on the initial task and the implementation. Possible applications are: given data about house attributes, such as room number, size, and price, predict the price of the previously unknown house; based on two datasets with healthy medical images and the ones with tumor, classify a pool of new images; cluster pictures of animals to several clusters from an unsorted pool. To develop a deeper understanding, it is worth going through the general workflow of the machine learning process, which is shown in Figure



As it can be seen, the process consists of 5 stages: 1. Data intake. At first, the dataset is loaded from the file and is saved in memory. 2. Data transformation. At this point, the data that was loaded at step 1 is transformed, cleared, and normalized to be suitable for the algorithm. Data is converted so that it lies in the same range, has the same format, etc. At this point feature extraction and selection, which are discussed further, are performed as well. In addition to that, the data is separated into sets – ‘training set’ and ‘test set’. Data from the training set is used to build the model, which is later evaluated using the test set. 14 3. Model Training. At this stage, a model is built using the selected algorithm. 4. Model Testing. The model that was built or trained during step 3 is tested using the test data set, and the produced result is used for building a new model, that would consider previous models, i.e. “learn” from them. 5. Model Deployment. At this stage, the best model is selected (either after the defined number of iteration or as soon as the needed result is achieved).

**Feature extraction**

In any of the examples mentioned above, we should be able to extract the attributes from the input data, so that it can be fed to the algorithm. For example, for the housing prices case, data could be represented as a multidimensional matrix, where each column represents an attribute and rows represent the numerical values for these attributes. In the image case, data can be represented as an RGB value of each pixel.

Such attributes are referred to as features, and the matrix is referred to as feature vector. The process of extracting data from the files is called feature extraction. The goal of feature extraction is to obtain a set of informative and non-redundant data. It is essential to understand that features should represent the important and relevant information about our dataset since without it we cannot make an accurate prediction. That is why feature extraction is often a non-obvious task, which requires a lot of testing and research. Moreover, it is very domain-specific, so general methods apply here poorly.

Another important requirement for a decent feature set is non-redundancy. Having redundant features i.e. features that outline the same information, as well as redundant information attributes, that are closely dependent on each other, can make the algorithm biased and, therefore, provide an inaccurate result.

In addition to that, if the input data is too big to be fed into the algorithm (has too many features), then it can be transformed to a reduced feature vector(vector, having a smaller number of features). The process of reducing the vector dimensions is referred to as feature selection. At the end of this process, we expect the selected features to outline the relevant information from the initial set so that it can be used instead of initial data without any accuracy loss.

Other possible transformations are:

1. **Normalization** An example of normalization can be dividing an image x, where xis are the number of pixels with color i, by the total number of counts to encode the distribution and remove the dependence on the size of the image. This translates into the formula: 𝑥 ′ = 𝑥 ||𝑥|| (Guyon and Elisseef 2006).

2. **Standardization** Sometimes, even while referring to comparable objects, features can have different scales. For example, consider the housing prices example. Here, feature ‘room size’ is an integer, probably not exceeding 5 and feature ‘house size’ is measured in square meters. Although both values can be compared, added, multiplied, etc., the result would be unreasonable before normalization. The following scaling is often done: x'i= (xi−µi)/σi , where µi and σi are the mean and the standard deviation of feature xi over training examples. (Guyon and Elisseef 2006).

3. **Non-linear expansions** Although in most cases we want to reduce the dimensionality of data, in some cases it might make sense to increase it. This can be useful for complex problems, where first-order interactions are not sufficient for accurate results.

**Supervised and Unsupervised Learning**

So far we have discussed the machine learning concepts from the point of view, where we have initial data, on which the model can be trained. However, this is not always the case. Here we want to introduce the two machine learning approaches - supervised and unsupervised learning.

In **Supervised Learning**, learning is based on labeled data. In this case, we have an initial dataset, where data samples are mapped to the correct outcome. The housing prices case is an example of supervised learning: here we have an initial dataset with houses, its attributes, and its prices. The model is trained on this dataset, where it ”knows” the correct results. Examples of supervised learning are regression and classification problems:

1. **Regression** Predict the value based on previous observations, i.e. values of the samples from the training set. Usually, we can say that if the output is a real number/is continuous, then it is a regression problem.

2. **Classification** Based on the set of labeled data, where each label defines a class, that the sample belongs to, we want to predict the class for the previously unknown sample. The set of possible outputs is finite and usually small. Generally, we can say that if the output is a discrete/categorical variable, then it is a classification problem.

In contrast to Supervised Learning, in Unsupervised Learning, there is no initial labeling of data. Here the goal is to find some pattern in the set of unsorted data, instead of predicting some value. A common subclass of Unsupervised Learning is Clustering:

3. **Clustering** Find the hidden patterns in the unlabeled data and separate it into clusters according to similarity. An example can be the discovery of different customer groups inside the customer base of the online shop.

**Classification methods**

From machine learning perspective, malware detection can be seen as a problem of classification or clusterization: unknown malware types should be clusterized into several clusters, based on certain properties, identified by the algorithm. On the other hand, having trained a model on the wide dataset of malicious and benign files, we can reduce this problem to classification. For known malware families, this problem can be narrowed down to classification only – having a limited set of classes, to one of which malware sample certainly 17 belongs, it is easier to identify the proper class, and the result would be more accurate than with clusterization algorithms. In this section, the theoretical background is given on all the methods used in this project.

**K-nearest neighbours**

K-Nearest Neighbors (KNN) is one of the simplest, though, accurate machine learning algorithms. KNN is a non-parametric algorithm, meaning that it does not make any assumptions about the data structure. In real world problems, data rarely obeys the general theoretical assumptions, making non-parametric algorithms a good solution for such problems. KNN model representation is as simple as the dataset – there is no learning required, the entire training set is stored. KNN can be used for both classification and regression problems. In both problems, the prediction is based on the k training instances that are closest to the input instance. In the KNN classification problem, the output would be a class, to which the input instance belongs, predicted by the majority vote of the k closest neighbors. In the regression problem, the output would be the property value, which is generally a mean value of the k nearest neighbors.

**Support Vector Machines**

Support Vector Machines (SVM) is another machine learning algorithm that is generally used for classification problems. The main idea relies on finding such a hyperplane, that would separate the classes in the best way. The term ’support vectors’ refers to the points lying closest to the hyperplane, that would change the hyperplane position if removed. The distance between the support vector and the hyperplane is referred to as margin. Intuitively, we understand that the further from the hyperplane our classes lie, the more accurate predictions we can make. That is why, although multiple hyperplanes can be found per problem, the goal of the SVM algorithm is to find such a hyperplane that would result in the maximum margins.

**Naive Bayes**

Naive Bayes is the classification machine learning algorithm that relies on the Bayes Theorem. It can be used for both binary and multi-class classification problems. The main point relies on the idea of treating each feature independently. Naive Bayes method evaluates the probability of each feature independently, regardless of any correlations, and makes the prediction based on the Bayes Theorem. That is why this method is called ”naive” – in real-world problems features often have some level of correlation between each other.

**Decision Tree**

As it implies from the name, decision trees are data structures that have a structure of the tree. The training dataset is used for the creation of the tree, that is subsequently used for making predictions on the test data. In this algorithm, the goal is to achieve the most accurate result with the least number of the decisions that must be made. Decision trees can be used for both classification and regression problems.

**Random Forest**

Random Forest is one of the most popular machine learning algorithms. It requires almost no data preparation and modeling but usually results in accurate results. Random Forests are based on the decision trees described in the previous section. More specifically, Random Forests are the collections of decision trees, producing a better prediction accuracy. That is why it is called a ’forest’ – it is basically a set of decision trees. The basic idea is to grow multiple decision trees based on the independent subsets of the dataset. At each node, n variables out of the feature set are selected randomly, and the best split on these variables is found.

**Modules and Algorithms**

**Scikit Learn**: Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

•**NumPy**: Base n-dimensional array package

•**SciPy**: Fundamental library for scientific computing

•**Matplotlib**: Comprehensive 2D/3D plotting

•**IPython**: Enhanced interactive console

•**Sympy**: Symbolic mathematics

•**Pandas**: Data structures and analysis

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

Although the interface is Python, c-libraries are leverage for performance such as numpy for arrays and matrix operations, LAPACK, LibSVM and the careful use of python.

**Algorithm**

**AdaBoost**, short for *Adaptive Boosting*, is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [meta-algorithm](https://en.wikipedia.org/wiki/Meta-algorithm) formulated by [Yoav Freund](https://en.wikipedia.org/wiki/Yoav_Freund) and [Robert Schapire](https://en.wikipedia.org/wiki/Robert_Schapire), who won the 2003 Gödel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and [outliers](https://en.wikipedia.org/wiki/Outlier). In some problems it can be less susceptible to the [overfitting](https://en.wikipedia.org/wiki/Overfitting_(machine_learning)) problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost (with [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) as the weak learners) is often referred to as the best out-of-the-box classifier.When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

AdaBoost is best used to boost the performance of decision trees on binary classification problems.

AdaBoost was originally called AdaBoost.M1 by the authors of the technique Freund and Schapire. More recently it may be referred to as discrete AdaBoost because it is used for classification rather than regression.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.

The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps.

Each instance in the training dataset is weighted. The initial weight is set to:

weight(xi) = 1/n

Where xi is the i’th training instance and n is the number of training instances.

**Hardware and Software Specification**

**Software Interface**

Client

* Interpreter : Python 3.6 or higher
* Libraries : Numpy, Scipy, Pandas, jupyter, sklearn, etc.,

Developer

* Operating system : Windows 7 or above or Linux
* IDE : Spyder or Pycharm
* Database : SQLite
* Interpreter : Python with Anaconda package
* Documentation tools : MS-word, MS-PowerPoint

**Hardware Interface**

Client

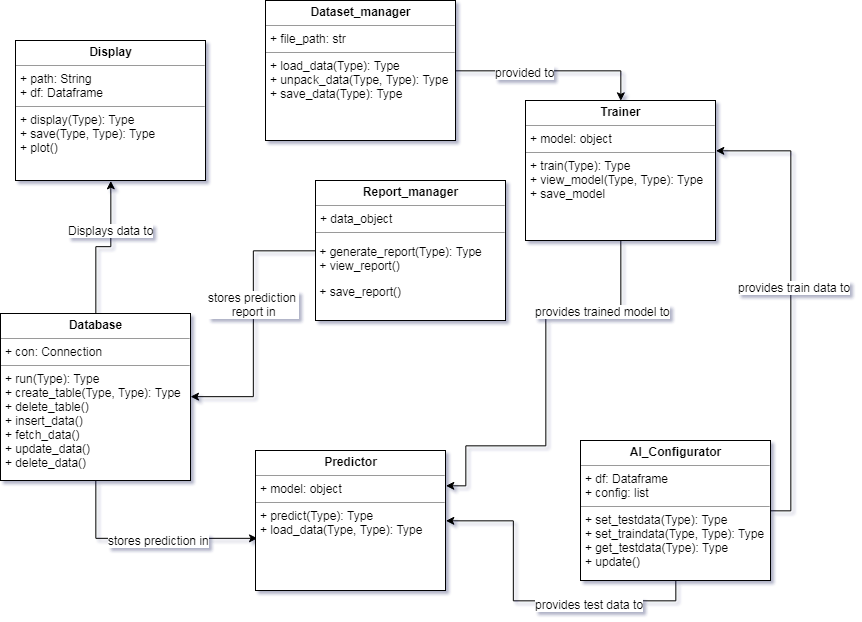
* + Processor : 1 GHz or above
  + RAM : 1 GB or higher

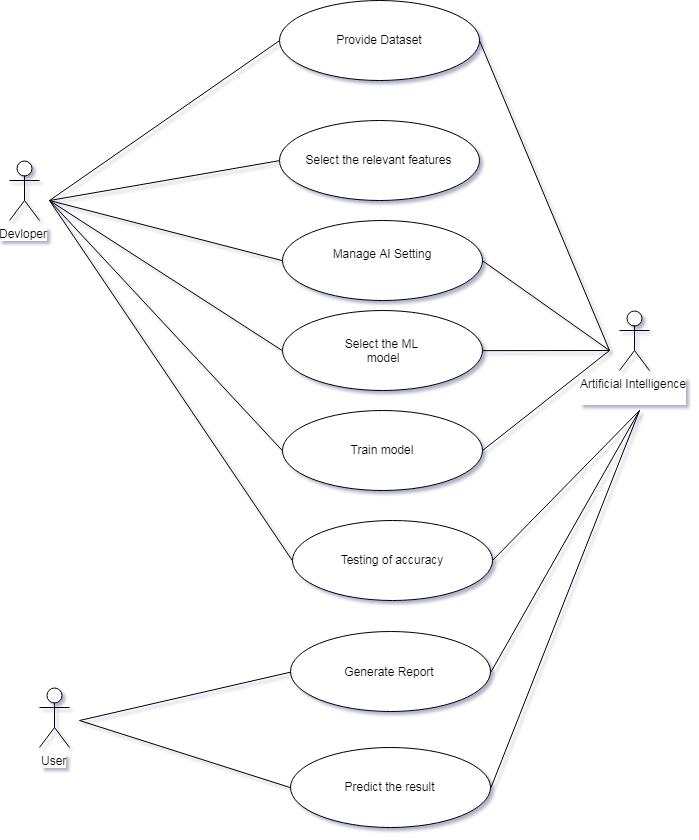
Developer

* + Processor : 2 GHz or above
  + RAM : 4 GB or higher

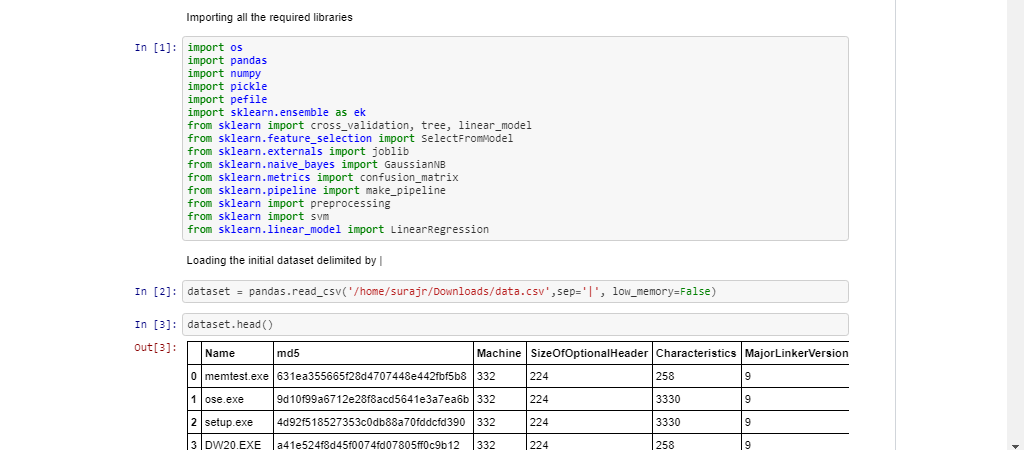
**Design**

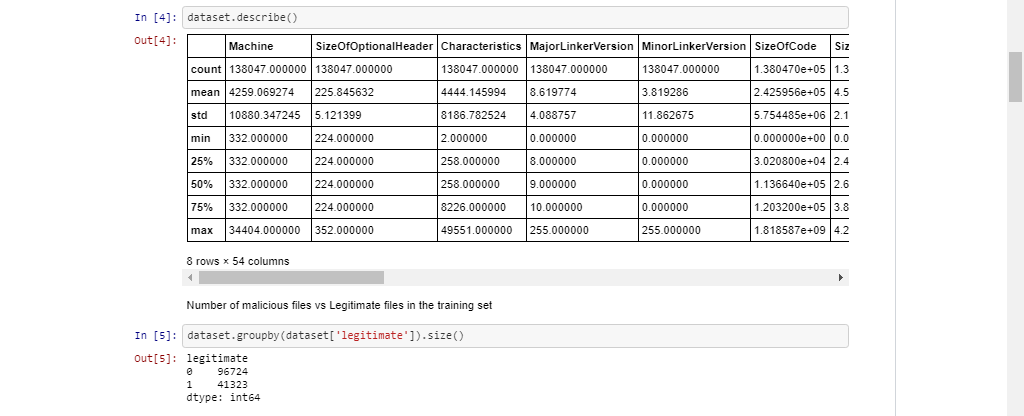
**UML Class Diagram**

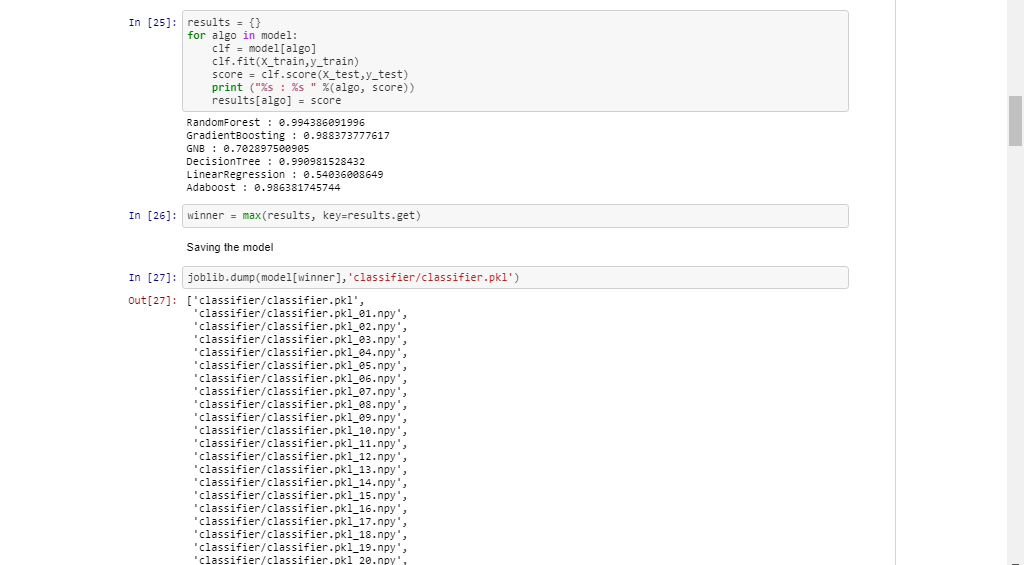
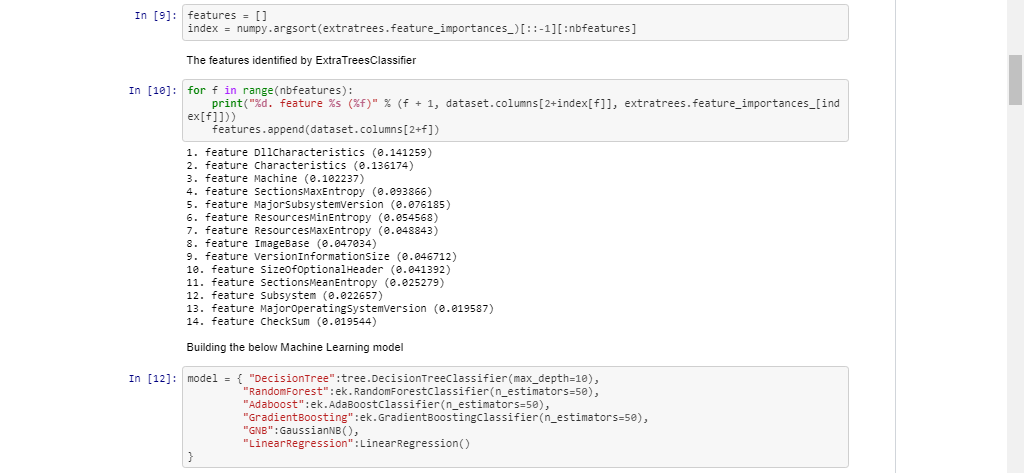


**UML Use Case Diagrams**

**Snapshots**

****





****

**Conclusion**

* Main target is to come up with a machine learning framework that generically detects as much malware samples as it can, with the tough constraint of having a zero false positive rate.
* Artificial intelligence cannot not automatically detect and resolve every potential malware or cyber threat incident, but when it combines the modelling of both bad and good behaviour, it can be a successful and powerful weapon against even the most advanced malware.
* Malware detection tools must constantly evolve to stay up to date with ever-changing crime ware.
* Although it’s difficult to imagine a world where human resources aren’t involved in combating evasive malware, the recent advances in AI will dramatically lessen our reliance on human assistance.

**References**

* [**www.darkreading.com**](http://www.darkreading.com)
* [**www.searchsecurity.techtarget.com**](http://www.searchsecurity.techtarget.com)
* [**www.github.com**](http://www.github.com)
* [**www.technical.nttsecurity.com**](http://www.technical.nttsecurity.com)
* **www.kaggle.com**