Bachelor thesis
Computing Science

Radboud University

How to DGA-based malware using BERT Transformer classifier

Author:
Abdulkarim Abdulkadir
s4840933

First supervisor:
Assistant Professor Katharina Kohls
`kkohls@cs.ru.nl`

[Second supervisor:]
title, name
`e-mail adress`

Second assessor:
title, name
`e-mail adress`

December 22, 2021

Abstract

The abstract of your thesis is a brief description of the research hypothesis, scientific context, motivation, and results. The preferred size of an abstract is one paragraph or one page of text.

Contents

Chapter 1

Introduction

As we increase to do all our tasks in the digital world, the importance of protecting our sensitive data is essential. The pandemic showed us how much we rely on this digital world. The value of our digital resources will thus also increase, making this a very interesting target for exploitation.

A malicious software, or malware, is a software that will cause damage to a computer system. The increase of the number of malware has been going rapidly. According to AV-Test report, in 2021 the number of malware totaled around 1250 million, a 12 times increase of that of 2012, which was around 100 million [1]. As the increase of the total amount of malware has increased, malware types and new attack methods are created and evolving by the day.

Most types of modern malware communicate with external servers using different network protocols, where DNS(Domain Name Server) services are used most frequently. The number of malicious domain names are also increasingly rapidly each year [2]. These malware frequently connect to a botnet, a network of computers running bots under control of the herder. The herder has a C & C(command and control) server that a infected computer(bot) is connected to and receives commands from.These malware often are using DNS Services to locate the C & C servers instead of using fixed IP addresses [3]. In particular, these malware uses DGA(domain generated algorithm) to generate domain names to avoid tracking. As a single domain or a fixed IP address can be easily tracked. These malicious domain names will then be put in a blacklist, which would make those domain names not accessible anymore. While malware that uses DGA connect to malicious domain names, which are active for a limited amount of time. This prevents them to being blocked by blacklist-based countermeasures. The domain names generated by the DGA are registered in advance to secure those generated domain names.

When a domain name is detected and blocked after an attack, the attacker can register another domain name that was generated by the DGA. Recently malware that uses DGA are polymorphic, even when this malware is analyzed and the DGA is examined, the malware can generate different domain names dynamically, by using for example time information as a seed for that algorithm [4].

There is a way needed to defend against DGA-based malware. Current defense techniques against DGA-based malware is using string information of the generated domain names and analyzing these string information. The attackers avoid this kind of defense techniques by creating DGA that is generating domain names that are almost not distinguishable from normal domain names [5]. To tackle this, there is a need for a new defense technique to differentiate the malicious based domain names from the unharmful ones. To solve this we will combine classification techniques and network traffic analysis.

Thus we can formulate our research question:
How to DGA-based malware using BERT Transformer classifier.

In chapter two we will explain how domain generated algorithms work and how it is implemented in a malware. As well as explaining the tools that are used in our experiment. We will also explain known network activity patterns in recent DGA-based malware and how these activities effect the system. In chapter three we will give a detail implementation and all technical details of our DGA detector/analyzer that analyzes, classifies and detect DGA-based malware samples. We will test our DGA detector/analyzer on recent DGA-based malware and unharmful programs that use DNS-based network traffic. In chapter four we will discuss previous research done in this field. In chapter five we will discuss and evaluate all our results and findings of the experiment. We will conclude in chapter six and give any suggestions for future research on this topic.

Chapter 2

Preliminaries

This section will describe malware, the different types and how it utilizes DGA perform malicious act. It will also explain the basics of machine learning and different types of neural networks that are necessary for this research paper to know.

Malware is a blending of two words, malicious and software, where it clearly defines the functionality of it, namely a software that is malicious in nature. Malware can have multiple purposes. Cybercriminals typically use it to extract data from the victims computer to leverage against them for financial gain. This data can range from financial data, sensitive personal data: such as healthcare records, personal emails, passwords, etc. The possibilities of the information that can be compromised are endless.

The most common ways victims receive malware is through the internet and mail.Malware can penetrate a victim's computer in different ways, such as: surfing to hacked websites, viewing malicious ads on websites, download infected files and install malicious programs or apps. When a malware has infected the computer system of a victim, it can come in many forms, such as Ransomware, Spyware, Trojans, Worms, etc.

2.1 Botnets

A compromised machine that is infected by malware can end up a network of infected machines (botnets). This machine is a bot in that network that receives and responds to commands from the command & control server. The C & C server is controlled by and receives commands by a human controller called a botmaster. The botmaster conceals itself by employing a number of proxy machines, called the stepping stones, between it and the C & C server. The life cycle of a botnet can be divided into four phases. For this research only the first two phases are important. The first phase is when the machine (bot) receives the malware and executes the binary. After the machine is infected, this machine (bot) tries to contact the C & C server to announce its presence and contact with it. This establishment phase is called Rallying. There are two ways that the bot can contact with the C & C server. The first way that the bot knows the IP address of the C & C server. This IP address can be hardcoded into the binary, which can be exposed by reverse engineering the binary. The IP address can also be seeded, where the bot is provided by a list of peers, this list can be hidden in Windows registries. The second way is that the bot knows the domain name of the C & C server. The domain name can be hardcoded into the bot binary, where it can resolve to different IP addresses. Reverse engineering this binary may expose the domain name, which can then be blacklisted.

2.2 Domain Generated Algorithm

The domain name can also be generated, then the bot dynamically contacts the C & C server using DGA (Domain name Generation Algorithm). The essence of this algorithm is that it creates a set of random strings. The bot attempts to resolve the generated domain names by sending DNS queries until one of the domains resolves to the C & C server IP Address. The domains that do not resolve will result in Non-Existent Domain (NXDomain) responses.

The domain names that are generated by the DGA are also known as Algorithmically Generated Domains (AGD). The DGA uses a seed to serve as a shared secret between the botmaster and the bot. There are two types of seeds: static seed and dynamic seed. The seed is required to calculate the AGDs. The DGA takes the seed value as input to generate pseudo-random strings and append algorithmically TLD (Top Level domains) such as *.nl, .com, .org, .edu.* The static seed can be dictionary of words, random strings that are concatenated, numbers or any other value that the botmaster can come up with. Dynamic seeds are dynamic, it changes with time. Dynamic seeds can be currency exchange rate, daily trending twitter hashtag, weather temperature and current date and time. The static and dynamic seed elements are then stitched together to generate a pseudo-random string.

The botmaster uses the DGA to generate a large number of domain names for the C & C server. The constant change of domain names for the C & C server using DGA is known as Domain-Fluxing. The botmaster registers one of the DGA created domain names for the C & C server in advance using the same algorithm of the DGA. When the bot receives the malware, the malware queries to the pre-registered domain name and resolves the IP address using DNS. The botmaster registers the domain name most of the time some hours prior to an attack and disposes of the domain names within a day. Whenever the previous domain name that the bot connected with does not resolve anymore, it queries to the next set of generated domain names until it find one domain that works.

The DGA and constant domain-fluxing of the C & C server provides agility and resilience to the infrastructure of the botmaster. This makes it hard to predict what domain names a bot will try. Analyst will re-engineer DGA by analyzing the malware and understand how the algorithm works. It is still hard to predict what kind of seed the DGA will use on a specific time. It is also infeasible to report all the domain names that are generated. As some DGA use english dictionary as static seed values, it is hard to distinguish benign domain names from malicious ones.

2.3 Machine Learning

Machine learning has recently been an attractive tool to be used in security. One way to combat DGA is to use machine learning to find the structure of the generated domains. The machine learning methods can be either supervised or unsupervised. Unsupervised learning uses algorithms to analyze and cluster data, in this case the domains. These algorithms discover hidden patterns or data groupings, without a need for a human intervention. There are three ways to approach unsupervised learning: clustering, association and dimensionality reduction. The domains are divided into clusters to find statistical attributes for each group. To produce a cluster with good generalization capabilities, it can take a lot of time and effort [2]. Supervised learning does not rely on the statistical attributes for each group to find DGAs. Supervised learning attempts to understand and classify the input and predict the outcome accurately. The relationship is represented as a structure to predict the outputs for some specific future inputs.

2.4 Neural Networks

Artificial Neural Networks are artificial systems that were inspired by the biological counterpart. The systems learn in a supervised manner to perform tasks by using various datasets and examples. These neural networks are composed of node layers, containing one or more input layers, hidden layers and output layers. Each node is connected to another node and has an associated weight w and threshold t. When the threshold of a specific node is above a certain threshold value, then that node is activated, otherwise no data is passed along to the next layer of the network. This is determined by a specificly used activation function in the network.

The network uses training data to learn and improve the accuracy of the network. This is usually done by backpropagation. Backpropagation is a supervised learning algorithm that computes the difference between the model output and the actual output using gradient descent and the chain rule . It checks if the error is minimized and update the weights and biases accordingly. It repeats the process until the error becomes minimum [3].

2.5 Activation Functions

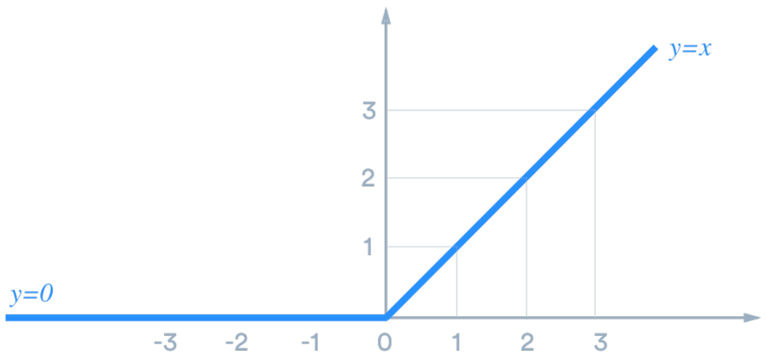
The activation functions in a neural network are functions that determine the output of a neural network. It maps it to a specific value. The function receives the calculated weighted sum of the inputs and the added bias and then decides if this passes through or not.

There are two types of activation functions: Linear Activation function and Non-linear Activation Functions. The linear activation functions are functions that does not change the weighted sum of the input, but instead returns the value directly. For multiple layer networks we need non-linear activation function.

2.5.1 ReLU

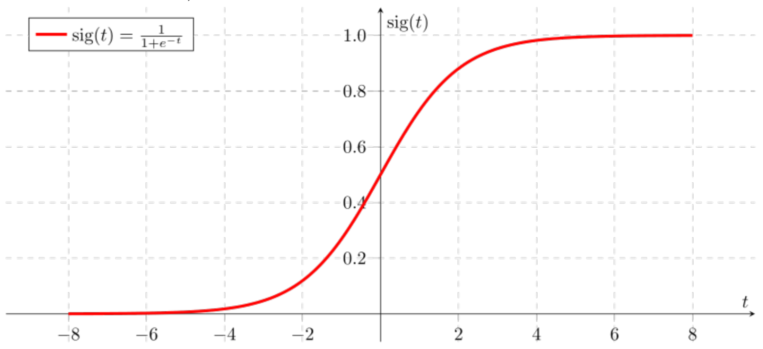
ReLU, or Rectified Linear Unit, is linear when the input is positive and 0 when the input is negative. The range of ReLu is [0, inf). The benefit of ReLu is that there is a reduced likelihood of the gradient to vanish(see subsection Vanish Gradient Problem), as the gradient is constant. This constant gradient results in faster learning of the network. Another benefit is the sparsity, as the network has more units in a layer, other activation functions will be processed to describe the output of that network. When the calculated sum that goes into the activation of ReLu is negative, it yields 0. This means there is fewer neurons firing, which makes the network lighter.

The disadvantage is that it tends to blow up, as the range goes to infinity and there is no mechanism to constrain the output when it is positive. Another disadvantage is the problem that if too many activations in the network get below zero, than the neurons in the network will output zero. This means that the outputs die out, thus prohibiting learning. This is called the Dying ReLu problem.



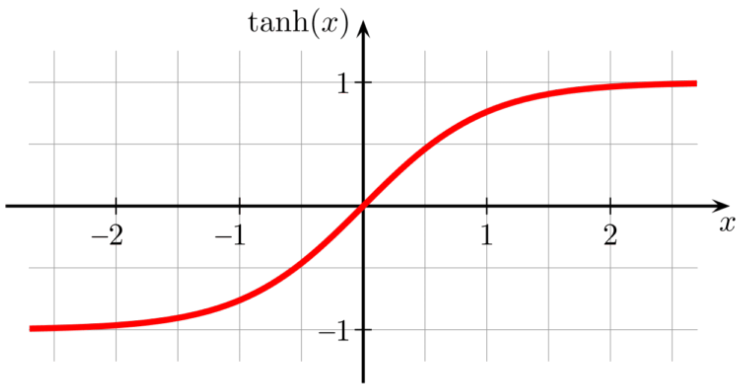
2.5.2 Sigmoid

Sigmoid activation function is a non-linear activation function that looks like an S-shape. Any small changes in the incoming X value(the calculated sum) will cause the Y value (the output) to change significantly. The range is [0,1], so it is bounded in a range and it does not blow up. The disadvantage of the Sigmoid function is the vanishing gradients(see subsection Vanishing Gradient Problem)



2.5.3 Tanh

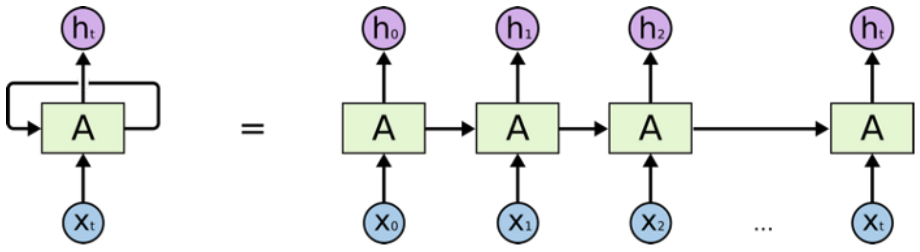
The Tanh activation function is very similar to the Sigmoid function. The difference is that the range is [-1,1] and the gradient is stronger for tanh than sigmoid. That means the derivative are steeper. One benefit of tanh over sigmoid is that it avoids bias in the gradients [4].



2.6 Recurrent Neural Networks

Recurrent neural networks or RNN, are a type of neural network that uses the output from the previous step and fed that as input in the current step, while in traditional Neural networks the network assumes that the inputs and outputs are independent of each other. The cost function or error can be calculated at any time t. At any given time t, the current input is a combination of x_t and x_{t-1}. This makes the neural network recurrent, it has feedback loops at each iteration of the hidden layer. One thing Recurrent Neural Networks are used for are Sequence Modeling. Sequence Modeling is the task to predict about future outcomes.

There are some drawbacks to RNN. The first drawback is that when the sequence is looked at in order, there will be a limit in how much parallelize training can be done. The second drawback is that the farther away the relevant points in the sequence are from one another, the harder and slower it is to make connections between them. This drawback is caused by the vanishing gradient problem.



2.6.1 Vanishing Gradient problem

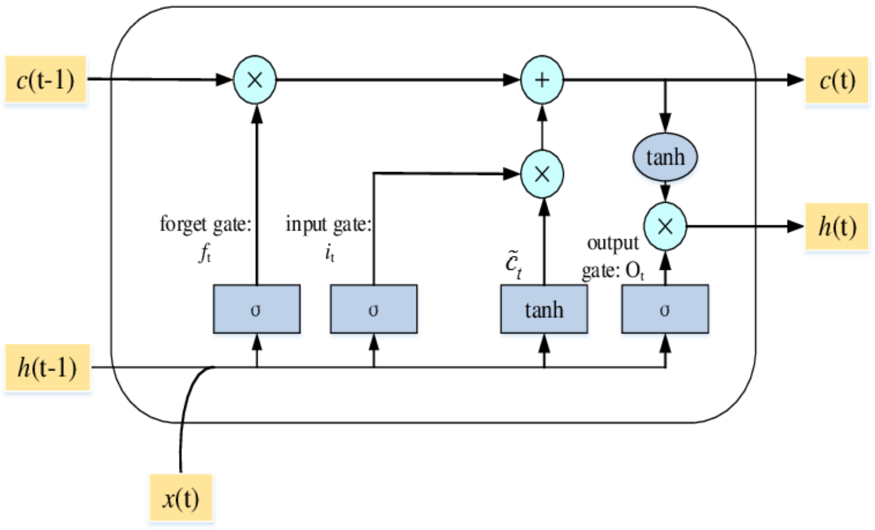
The vanishing gradient problem can be encountered when gradient-based learning methods and backpropagation is used. As there are more layers using activation function are added to the neural network, the gradients of these loss functions approaches zero. The gradient will be vanishingly small, which in turn prevents the current weight from changing its value. This can lead to the neural network to stop further training. As mentioned before, an activation function like the sigmoid function, squishes a large input space into a value between 0 and 1. The effect of this is that a large change in input would cause a small change in the output. The derivative therefore becomes very small. The derivative approaches zero, which causes the gradient of this layer in the network to vanish.

As mentioned above, one network that suffers from vanishing gradient problem is a traditional Recurrent Neural Network. When the feedback loops occurs and the gradient is getting lower, then it is more harder for the network to update it weights. The weights of the initial value will not change effectively through the training process, which can lead to inaccuracy in the network.

The solution to the vanishing gradient problem is to use other activation functions for the network, such as ReLu, which doesn't cause a small derivative. Another solution is to use residual networks [1]. There are also specialized RNNs, that are resistant to this problem. One of these architectures is the Long short-term memory (LSTM) network.

2.6.2 LSTM

Long short-term memory (LSTM) is a specialized RNN, that is capable of learning in long-term dependencies. It is designed to remember information for long periods of time. It does that by adding a forget mechanism. The hidden layer in LSTM is a gated cell. It consists of four layers that interact with each other to produce the output of that cell to pass on to the next hidden layer. LSTM is compromised of three logistic sigmoid gates and one tanh layer, compared to traditional RNNs that use only single layer of tanh. These gates are used in order to limit information or pass information through the cell. The inputs of LSTM go through the input gate, forget gate and output gate. The forget gate decides to remember or to skip inputs from the previous hidden states. The mechanism of the forget gate mostly solves the vanishing gradient problem. The input gate decides what new information has to be added to the cell. Finally the output gate decides which new information or old information has to be passed to the next hidden layer by using the memory state that are updated by the input and forget gate.



While LSTM overcame the vanishing gradients in the network, it still inherits some problems of RNNs, such as: no parallelization, as there is still a sequential path for the data. While LSTMs was created to solve the vanishing gradient problem, it could not remove this problem completely. This is because the data still has to move from cell to cell in a sequential manner. The problem inherently lies in the recursion of RNNs.

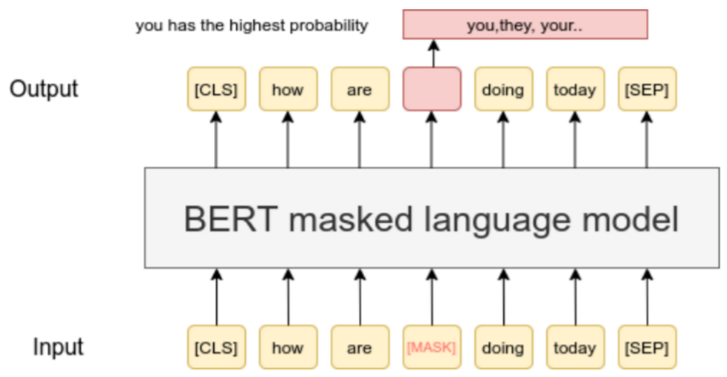
2.7 Transformers

A transformer is a new deep learning model that uses a mechanism called self-attention

2.7.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a large transformer masked language model. It is mostly used for pre-training natural language processing (NLP). The main innovation of this technique is applying bidirectional training on a Transformer model. In contrast, other efforts looked at it in a single direction, from left to right or right to left.

BERT only uses the encoder mechanism to generate a language model. The encoder of BERT reads the entire sequence of words at once. That ensures that the model learns the context of a word from all its surroundings, thus making it bidirectional. BERT are pretrained on two tasks: language modelling (LM) and next sequence prediction (NSP). It uses Masked LM (MLM) for pretraining language modelling. This is done by replacing 15% of any sequence with a [MASK] token. The model tries to predict the original value of the masked words, using the context provided by the other non-masked words in a sequence. Masked language models are a kind of contextual word embedding models. Contextual word embedding gives a model different representation for different sentences.



The next task of the training process, BERT uses next sentence prediction to understand the relationship between two sentences. The model receives as input pairs sentences and it learns to predict the second sentence is the next sentence in the original document. BERT separates sentences with a special [SEP] token. Then the model is fed with two input sentences at a time. During training, 50% of the time the second sentence is the subsequent sentence in the original document, while in the other 50% of the time it is a random sentence from the full corpus. The assumption being that the random sentence will be disconnected from the first sentence. All together the input in BERT is processed in these steps:

1. Each sentence sequence is separated by a [SEP] token. It is placed at the end of each sentence.
2. Every sentence will replace 15% of its words with a [MASK] token.
3. At the beginning of the first sentence a special [CLS] token will be inserted.
4. Every other word in a sentence is transformed into an embedded token.
5. A sentence embedding is added to each token. They are similar in concept as token embeddings, but these embeddings are used to indicate if sentence A or sentence B is added to each token.
6. At last a positional embedding gets added to every token to indicate their position in the sequence.

Chapter 3

Research

This chapter, or series of chapters, delves into all technical details that are required to *prove* your scientific hypothesis. It should be sufficiently detailed and precise in order for any fellow computing scientist student to be able to *repeat* your research and therewith establish the same results / conclusions that you have obtained. Please note that, in order to improve readability of your thesis, you can put a part of this information also in one or more appendices (see Appendix A).

Chapter 4

Related Work

In this chapter you demonstrate that you are sufficiently aware of the state-of-art knowledge of the problem domain that you have investigated as well as demonstrating that you have found a *new* solution / approach / method.

Chapter 5

Conclusions

In this chapter you present all conclusions that can be drawn from the preceding chapters. It should not introduce new experiments, theories, investigations, etc.: these should have been written down earlier in the thesis. Therefore, conclusions can be brief and to the point.

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] S. Krishnan, F. Monrose, and J. Mchugh. Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing. *43 Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 1–12, 2013.

[3] Claude Lemarechal. *Cauchy and the Gradient Method*, 2010. `https://www.math.uni-bielefeld.de/documenta/vol-ismp/40_lemarechal-claude.pdf`.

[4] Leon Bottou Yann LeCun, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. *Image Processing Research Department AT&T Labs*, pages 1–12, 1998.

Appendix A

Appendix

Appendices are *optional* chapters in which you cover additional material that is required to support your hypothesis, experiments, measurements, conclusions, etc. that would otherwise clutter the presentation of your research.