Adversarial Attacks on Transformers-Based Malware Detectors

Yash Jakhotiya
College of Engineering, Pune
jakhotiyays16.comp@coep.ac.in

Heramb Patil
College of Engineering, Pune
herambnp16.comp@coep.ac.in

Jugal Rawlani
College of Engineering, Pune
rawlanijr16.comp@coep.ac.in

Abstract

Signature-based malware detectors have proven to be insufficient as even a small change in malignant executable code can bypass these signature-based detectors. Many machine learning-based models have been proposed to efficiently detect a wide variety of malware. Many of these models are found to be susceptible to adversarial attacks - attacks that work by generating intentionally designed inputs that can force these models to misclassify. Our work aims to explore vulnerabilities in the current state of the art malware detectors to adversarial attacks. We train a Transformers-based malware detector, carry out adversarial attacks resulting in a misclassification rate of 23.9% and propose defenses that reduce this misclassification rate to half. An implementation of our work can be found at https://github.com/yashjakhotiya/Adversarial-Attacks-On-Transformers.

1 Introduction

1.1 Malware

Malware is software written to steal credentials of computer users, damage computer systems, or encrypt documents for ransom, among other nefarious goals. With close to two-thirds of the world population connected to the internet, malware programs pose serious problems to this growing number of netizens [Kemp, 2022]. In Q1 of 2021 alone, around 87.6 million new types of malware and 2.51 million new types of ransomware were detected, summing the total number of malware detected till 2021 to more than 1.51 billion and these figures keep growing constantly [Beek et al., 2021].

1.2 Malware detectors

Malware detection thus becomes an integral part of today's internet infrastructure. One of the major problems faced by the malware detectors community is detecting the sheer number of malware that has been newly created and not been analysed before. A prevalent way used in commercial antivirus products is using signature-based malware detection with signatures extracted by expert analysts, with a small room for malware variants. These signature-based approaches have some serious limitations. First, it is easy to bypass such detectors by generating a seemingly different executable code than a known malware, but with the same functionality, using a different compiler. Second, specialized obfuscation techniques too exist, which apply transformations that evade these malware detectors [Canfora et al., 2015]. Third, the approach of manually generating signatures for new malware is unscalable with the rapid addition of new unseen types of malware. Thus, there is a dire need to create intelligent malware detection systems that automatically derive features from malware executables

Preprint. Under review.

that are generalizable enough to counter current obfuscation techniques and can extend to new types of malware. For this purpose, many machine learning-based malware analysis methods have been proposed [Schultz et al., 2001] [Kolter and Maloof, 2004] [Dai et al., 2009] [Baldangombo et al., 2013].

1.3 Deep learning in malware detection

These machine learning-based approaches work by deriving static features to categorize malware. A serious advantage these models have over signature-based methods is that they do not require domain experts to manually analyze and create signatures for the large body of new malware, thus saving extensive human labour. However, focusing only on static features may not represent the full semantic meaning of an executable [Aghakhani et al., 2020].

Deep learning-based approaches that can automatically learn representational feature space mappings from malware executable code have been proposed in an effort to have better generalizability than hand-crafted features used in signature-based and static features used in classic machine learning methods [Saxe and Berlin, 2015] [Kalash et al., 2018] [Tobiyama et al., 2016]. However, the long sequential nature of executables makes it difficult for the popular feedforward, CNN and RNN-based deep learning architectures to fully capture the semantic meaning of an executable file.

In Sept 2017, the Transformer, a self-attention mechanism-based neural network architecture was introduced which performed better than contemporary convolutional as well as recurrent models on academic English to French and English to German translation benchmarks [Vaswani et al., 2017]. In addition to the greater performance on translation, the Transformer requires less computational runs to train and can be considered better support for the underlying machine learning hardware, along with increasing training speeds by many folds [Vaswani et al., 2017]. Due to similar semantic structures present in natural languages and assembly languages, Transformers-based malware detectors are predisposed to overcome the above major limitations of previous neural network architectures for malware detection.

1.4 Deep learning and adversarial attacks

Deep learning is being used for solving a variety of complex problems and it has provided a compact framework for dealing with problems that were difficult to solve using traditional machine learning techniques [Xiong et al., 2015] [Amodio et al., 2019] [Huang et al., 2017]. In recent years, the advancement in deep learning has enabled it to provide performance at par with what humans can do on several tasks [Silver et al., 2017] resulting in growing faith in such real world deployed systems [Tesla, 2020] [Apple, 2020] [Grigorescu et al., 2020]. On the other hand, deep learning systems are found to be vulnerable to different generated adversarial attacks [Szegedy et al., 2013], which are malicious inputs specially designed to confuse a trained model to wrongly classify the output. Recently, several researchers have been successful in generating different kinds of adversarial attacks by leveraging vulnerabilities after analysing the threat model to eventually compromise the model [Tu et al., 2020] [Chen et al., 2020].

2 Related Work

Deep neural network-based techniques for malware detection Rule-based signature-based approaches require a cybersecurity researcher to manually set up rules, or categorize a binary as malware and mark its signature. This would require researchers to know how every new malware works and is not a scalable approach. [Saxe and Berlin, 2015] propose a deep learning based approach to help solve this problem.

Adversarial attacks on deep neural network-based malware detectors [Stokes et al., 2017] describe using deep learning for malware detection as a double-edged sword, where deep learning could be really helpful in identifying new, yet unknown malware, but miscreants can also come up with ways to fool the neural networks by creating something known as an adversarial sample which are small perturbations that do not change the sample's original function, but rather fools the network into classifying it into some other class.

CNN-based neural networks for malware detection [Kalash et al., 2018] used CNNs to classify binaries as malware or benign files where binaries converted to an image representation were used. The authors were able to achieve best accuracy of 98.52% for the Malimg dataset [Nataraj et al., 2011], and best accuracy of 98.99% for the Microsoft Malware Dataset [Ronen et al., 2018].

Adversarial attacks on CNN-based malware detectors Given that CNNs are also vulnerable to adversarial attacks, [Chen et al., 2019] evaluated various methods of conducting adversarial attacks on CNN based malware detectors. They used a dataset of 5198 malicious files, and 5200 benign files from a clean installation of Windows. The best accuracy of the model was around 94% in classifying files as malicious or benign. The success rate of white-box attacks for the Fast Gradient Sign Method (FGSM) was really low around 3%, whereas for the Bit-Flip Attack (BFA) it was around a mean of 20%. Black blox methods did not perform that well, the random method had a mean success rate of around 4%, and the Experience based method had a mean success rate of 45%.

RNN-based neural networks for malware detection After the success which recurrent neural networks have shown for other tasks, they have been tried for the task of malware detection [Beek et al., 2021]. [Tobiyama et al., 2016] used a combination of convolutional neural networks and recurrent neural networks for the purpose of malware detection. RNNs were used for feature extraction and CNNs were used for feature classification. They obtain a best case AUC score of 0.96.

Adversarial attacks on RNN-based malware detectors With the use of RNN for malware detection, it became known that even they are susceptible to adversarial samples due to the general susceptibility of neural networks to adversarial attacks [Hu and Tan, 2017]. Generally to an attacker, the architecture of the network used for malware detection is not known and access to model weights is not present. To simulate this black-box nature of the model, [Hu and Tan, 2017] first trained a substitute RNN to simulate the behavior of the detector to be attacked. Another RNN was trained to create adversarial samples from malware inputs. The first RNN is called Substitute RNN while the latter is known as Generative RNN.

Transformer-based neural networks for malware detection As it came to light that malware detectors based on malware signatures were failing to capture novel malware, many machine learning methods were proposed as explained in previous sections. These methods did not look at the whole meaning of the assembly code, but rather looked at different chunks of the assembly language instructions. To overcome this, Transformer-based neural networks for malware detection were proposed by [Li et al., 2021]. These Transformer-based approaches achieve better accuracy than previous approaches ([Moskovitch et al., 2008], [Baldangombo et al., 2013], [Saxe and Berlin, 2015], [Mourtaji et al., 2019]) in all experiments. Additionally, the model achieves never seen before interpretability.

3 Training a Transformer for malware detection

In this section we list down the details of training a competitive Transformers-based malware detector on which we will carry out an adversarial attack in section 5, and evaluate defenses against the attack in section 6.

3.1 System Design and Architecture

Our malware detection system is mainly divided into 3 parts:

1. **Assembly Module** The assembly module consists of a disassembler, a tokenizer and a Transformer. The input to the assembly module is an exe file, which is fed directly to the disassembler. The assembly module is responsible to calculate assembly language features, which would be used for final classification
2. **Static Feature Module** The static feature module consists of a DLL extractor, and a string extractor. The input to this is the same as that to the assembly module, an exe file. The DLL extractor extracts PE imports from the file, and the string extractor extracts all the printable strings from the given input file. The static feature module outputs two set of vectors, one

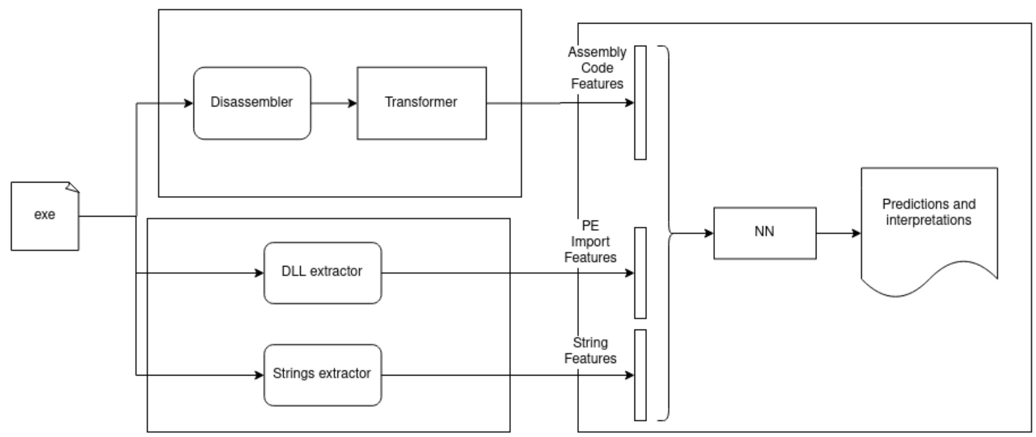

Figure 1: System Architecture

from the DLL extractor, and the other from string extractor. The output from the static feature module will be used for final classification.

3. **Neural Network Module** The neural network module consists of a neural network, which takes in assembly language features from the assembly module, and PE import features and string features from the static feature module, and performs a binary classification on whether the file is malicious or benign.

3.2 Transformer Architecture

A transformer is basically built of multiple building blocks called encoders and decoders. The encoder takes in a sequence of data, and outputs a vector representation of all the data given as input. The decoder, being the opposite of the encoder, takes in a vector input and produces sequential data as its output.

The output from the disassembler is fed to the encoder after tokenization, which then produces a vector semantically representing all the instructions fed. This is helpful because the complete instruction set cannot be used as a feature for the final feed-forward neural network as that would be unscalable and will have different dimensions for different input binaries.

3.3 Static Features

The feed-forward NN also takes input static features extracted from the PE binaries. The static features include :

- **PE Imports :** PE Imports are the DLL files imported by the given PE binary. This helps to capture the external function calls, and imports, and hence helps in categorizing suspicious files based on import patterns from the existing malware binaries [Saxe and Berlin, 2015].
- **Printable strings :** All printable strings (only ASCII characters) of size greater than or equal to 6 are extracted from the given binary and used as another feature to train the vanilla NN.

3.4 Feature Set

The final feature set is thus formed by combining (1) the feature vector obtained from the Transformer, (2) the DLL feature vector obtained from Static Feature Extractor, and (3) the string feature vector obtained from Static Feature Extractor. The final feature vector is then fed to the fully connected neural network for classifying files as malicious or benign.

3.5 Dataset Preparation

We performed our expermients by collecting a total of 2985 malware samples from VirusTotal and 2215 benign executables from a fresh Windows 10 installation. We used objdump to disassemble binary executables into .asm files. and tokenized strings to create a vocabulary which was eventually fed to the transformer.

3.6 Detection Results

We were able to classify malware and benign files with a test set accuracy of 92.5%.

4 Adversarial attacks

4.1 Introduction

Machine learning techniques are generally designed to work on a specific statistical distribution with training and evaluation being performed on the same distribution of inputs. Models trained in this way are vulnerable to misclassification when provided with a set of inputs that come from a different distribution. These inputs can be modified with special techniques to force misclassification. Such generated inputs are known as adversarial examples and the fooling of a machine learning model with such adversarial examples is known as carrying out an adversarial attack.

Notable examples in the past that have successfully demonstrated adversarial attacks include McAfee's fooling of Tesla autonomous vehicle by just adding 2 black strips on a speed limit sign making the Tesla AV go 50 miles per hour past the limit [Barrett, 2020]. Dresswear that can fool face detection systems or license plates that can fool automatic license number capture systems use adversarial examples to make such models misclassify [Seabrook, 2020].

4.2 Classification of attacks

Adversarial attacks can be classified into two broad types of Evasion attacks and Poisoning attacks. An evasion attack is used when the attacker does not have access to the model during the training phase. This is the most widely used attack where input is tweaked to force the model into misclassification during the testing phase. On the other hand, a poisoning attack compromises the model in the training phase itself by giving it inputs that are malicious.

In this work, we consider evasion attacks as we consider that to be a generalized case where an attacker does not have access to the training of the model. Evasion attacks can be broadly divided into the two steps of estimation of sensitivity for each direction of perturbation and selection of directions for perturbation. In step one, the attacker determines the amount of perturbation to add to in each direction which can most likely make the model misclassify the input. In step two, the attacker either adds these perturbations in all directions or chooses those directions that can make the model misclassify most likely.

For the estimation of sensitivity, there are multiple algorithms available. Some of them are listed below -

- **L-BFGS -** Introduced by [Szegedy et al., 2013], this method tries to solve the minimization problem of finding the minimum perturbation that can force misclassification with the L-BFGS optimization method.

$$arg\ min\ _r\ f(x+r) = l \qquad (x+r) \in D \tag{1}$$

  where l is not equal to the target label $h(x)$.

- **FGSM -** [Goodfellow et al., 2014] introduced the Fast Gradient Sign Method that can solve the equation above computationally efficiently.

$$X^* = X + \varepsilon * sign(\nabla_x C(X, Y_{true}) \tag{2}$$

  Here C is the cost function used in the model. X^* is the adversarial counterpart of the input X. ε is the amount of perturbation. ∇_x is the gradient of the cost function.

- **Jaccobian Method -** [Papernot et al., 2015] determined sensitivity in each dimension by finding out Jacobian of the trained model, i.e. it's derivative in a forward way and then perturbed input in most sensitive dimensions.

5 Attacking our trained Transformer

We used the Fast Gradient Sign Method by [Goodfellow et al., 2014] to craft out adversarial samples to fool the Transformer-based malware detector trained in section 3. FGSM can be implemented in two ways, either by using the target class directly or by using the iterative method. We used the target class directly to generate our adversarial samples by substituting Y_{true} with Y_{target} in equation 2 above.

$$X^* = X + \varepsilon * sign(\nabla_x C(X, Y_{target}) \quad (3)$$

We perturbed all possible dimensions and achieved a misclassification rate of 23.9% with the Fast Gradient Sign Method.

6 Defenses against adversarial attacks

With the increase in adversarial attacks on commercial systems, many defenses have been proposed against them. Although these defenses do not provide complete immunity against adversarial samples, they act as deterrents. Some common and new defenses are listed below.

- **Training on adversarial samples -** This is a brute force approach where the input distribution of the model is expanded. Although this defense is not very useful in the case of black-box attacks as shown by [Narodytska and Kasiviswanathan, 2017], it is still widely used as a practical defensive approach against adversarial samples.
- **Masking the gradient of the model -** Many attack methods including FGSM depend on the derivative of the trained model. Nearest neighbor classifiers or decision trees-like models can effectively deter such an adversarial attack. However, these methods often underperform when compared to state of the art neural architectural methods.
- **Reducing feature space -** Developed by [Xu et al., 2018], this method aims at reducing the feature space, also known as feature squeezing to make step one of generating adversarial samples discussed above difficult. With less number of features, the number of dimensions to add perturbations to declines naturally. To achieve the same effect as shown by higher dimension inputs, stronger perturbations need to be added which can break the original input's purpose.
- **Transferability block -** Adversarial attacks are successful in many cases due to the transferability property of neural networks. Transferability property says that adversarial samples generated on one neural network can make other neural networks misclassify even if they are trained on inputs that come from different data distributions and are made up of inherently different architectures. [Hosseini et al., 2017] block this transferability of neural architectures by training models to predict a NULL label to peturbed inputs.

In our approach, we set up defenses for our model with two defenses from the ones listed above. With the most practical adversarial training, the misclassification rate dropped to 11.2%. With reducing the feature space, we did not get promising results, and the misclassification rate reduced by a mere 2.4% to 21.5%.

7 Conclusion and Future Scope

The use of deep learning techniques for the task of malware detection has given promising results and is in use at a few of the most sought after anti-malware products [Kaspersky Enterprise Cybersecurity, 2017]. But due to the inherent nature of such deep learning techniques, these malware detectors are prone to adversarial attacks. We have implemented an avant-garde machine learning detector using Google's Transformer neural network architecture, demonstrated an adversarial attack on the same, and proposed defenses against such adversarial attacks. The future scope in this directoin could aim at demonstrating more types of adversarial attacks on such malware detectors and propose better defenses that do not need access to the trained model.

References

H. Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. When malware is packin' heat; limits of machine learning classifiers based on static analysis features. In *NDSS*, 2020.

Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature methods*, pages 1–7, 2019.

Apple. About face id advanced technology. `https://support.apple.com/en-au/HT208108`, 2020.

Usukhbayar Baldangombo, Nyamjav Jambaljav, and Shi-Jinn Horng. A static malware detection system using data mining methods. *CoRR*, abs/1308.2831, 2013. URL `http://arxiv.org/abs/1308.2831`.

Brian Barrett. A tiny piece of tape tricked teslas into speeding up 50 mph | wired. `https://www.wired.com/story/tesla-speed-up-adversarial-example-mgm-breach-ransomware/`, 2020.

Christiaan Beek, Mo Cashman, John Fokker, Melissa Gaffney, Steve Grobman, Tim Hux, Niamh Minihane, Lee Munson, Chris Palm, Tim Polzer, Thomas Roccia, Raj Samani, and Craig Schmugar. Mcafee labs threats report, june 2021. `https://www.mcafee.com/enterprise/en-us/assets/reports/rp-threats-jun-2021.pdf`, 2021.

Gerardo Canfora, Andrea Di Sorbo, Francesco Mercaldo, and Corrado Aaron Visaggio. Obfuscation techniques against signature-based detection: A case study. In *2015 Mobile Systems Technologies Workshop (MST)*, pages 21–26, 2015. doi: 10.1109/MST.2015.8.

Bingcai Chen, Zhongru Ren, Chao Yu, Iftikhar Hussain, and Jintao Liu. Adversarial examples for cnn-based malware detectors. *IEEE Access*, 7:54360–54371, 2019. doi: 10.1109/ACCESS.2019.2913439.

Xuesong Chen, Xiyu Yan, Feng Zheng, Yong Jiang, Shu-Tao Xia, Yong Zhao, and Rongrong Ji. One-shot adversarial attacks on visual tracking with dual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10176–10185, 2020.

Jianyong Dai, Ratan K Guha, and Joohan Lee. Efficient virus detection using dynamic instruction sequences. *J. Comput.*, 4(5):405–414, 2009.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014. URL `https://arxiv.org/abs/1412.6572`.

Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems, 2017. URL `https://arxiv.org/abs/1703.04318`.

Weiwei Hu and Ying Tan. Black-box attacks against RNN based malware detection algorithms. *CoRR*, abs/1705.08131, 2017. URL `http://arxiv.org/abs/1705.08131`.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil D. B. Bruce, Yang Wang, and Farkhund Iqbal. Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2018. doi: 10.1109/NTMS.2018.8328749.

Kaspersky Enterprise Cybersecurity. Machine learning for malware detection. `https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf`, 2017.

Simon Kemp. The global state of digital in july 2022 — datareportal – global digital insights. `https://datareportal.com/reports/digital-2022-july-global-statshot`, 2022.

Jeremy Z. Kolter and Marcus A. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 470–478, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138881. doi: 10.1145/1014052.1014105. URL `https://doi.org/10.1145/1014052.1014105`.

Miles Q. Li, Benjamin C.M. Fung, Philippe Charland, and Steven H.H. Ding. I-MAD: Interpretable malware detector using galaxy transformer. *Computers & Security*, 108:102371, sep 2021. doi: 10.1016/j.cose.2021.102371. URL `https://doi.org/10.1016%2Fj.cose.2021.102371`.

Robert Moskovitch, Clint Feher, Nir Tzachar, Eugene Berger, Marina Gitelman, Shlomi Dolev, and Yuval Elovici. Unknown malcode detection using opcode representation. In Daniel Ortiz-Arroyo, Henrik Legind Larsen, Daniel Dajun Zeng, David Hicks, and Gerhard Wagner, editors, *Intelligence and Security Informatics*, pages 204–215, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-89900-6.

Youness Mourtaji, Mohammed Bouhorma, and Daniyal Alghazzawi. Intelligent framework for malware detection with convolutional neural network. In *Proceedings of the 2nd International Conference on Networking, Information Systems Security*, NISS19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366458. doi: 10.1145/3320326.3320333. URL `https://doi.org/10.1145/3320326.3320333`.

Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, 2017. doi: 10.1109/CVPRW.2017.172.

L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. Malware images: Visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306799. doi: 10.1145/2016904.2016908. URL `https://doi.org/10.1145/2016904.2016908`.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings, 2015. URL `https://arxiv.org/abs/1511.07528`.

Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. Microsoft malware classification challenge. *CoRR*, abs/1802.10135, 2018. URL `http://arxiv.org/abs/1802.10135`.

Joshua Saxe and Konstantin Berlin. Deep neural network based malware detection using two dimensional binary program features. *CoRR*, abs/1508.03096, 2015. URL `http://arxiv.org/abs/1508.03096`.

Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and S. Stolfo. Data mining methods for detection of new malicious executables. *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, pages 38–49, 2001.

John Seabrook. Dressing for the surveillance age | the new yorker. `https://www.newyorker.com/magazine/2020/03/16/dressing-for-the-surveillance-age`, 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Jack W. Stokes, De Wang, Mady Marinescu, Marc Marino, and Brian Bussone. Attack and defense of dynamic analysis-based, adversarial neural malware classification models. *CoRR*, abs/1712.05919, 2017. URL `http://arxiv.org/abs/1712.05919`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013. URL `https://arxiv.org/abs/1312.6199`.

Tesla. Future of driving, 2020. `https://www.tesla.com/en_AU/autopilot`.

Shun Tobiyama, Yukiko Yamaguchi, Hajime Shimada, Tomonori Ikuse, and Takeshi Yagi. Malware detection with deep neural network using process behavior. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 577–582, 2016. doi: 10.1109/COMPSAC.2016.151.

James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Hui Y Xiong, Babak Alipanahi, Leo J Lee, Hannes Bretschneider, Daniele Merico, Ryan KC Yuen, Yimin Hua, Serge Gueroussov, Hamed S Najafabadi, Timothy R Hughes, et al. The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218), 2015.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018. doi: 10.14722/ndss.2018.23198. URL `https://doi.org/10.14722%2Fndss.2018.23198`.