Chapter 1

State of the Art

There have been multiple approaches to obfuscate malware in the past using different machine learning methods. The following chapter will outline the most promising approaches regarding their results and reproducibility.

1.1 Evasion of Machine Learning Models

An attacker has a variety of methods to evade machine learning models. Prominent methods rely on the following issues:

- Concept drift referrs to a change of the data distribution since the model was trained. This effects supervised learning algorithms where the input instances as well as the target value shift over time. In traditional offline learning this results in the need to retrain the whole model [18].
- The modeling error describes a case where a machine learning model uses $p'(x|y)$ which is a miscalculated version of the true posterior probabilty $p(x|y)$. This enables an attacker to find a variant x' of x that would lead to $p(x'|y) > \eta > p'(x'|y)$ where η is some chosen threshold [4].
- The Bayes error rate [8] is a lower bound of the error rate that exists even in models where $p(x|y) = p'(x|y)$. It classifies the irreducible error of a task corresponding to the error rate of a bayes optimal classifier. This error rate is greater than zero whenever overlaps between $p(x|y(malicious))$ and $p(x|y(benign))$ exist [4]. This is a fundamental issue in antivirus products where the tradeoff between false positives and false negatives has to be adressed. The treadeoff is mostly beeing shifted towards false negatvies to avoid a great number of alerts due to false positives on a customer system [15].

1.2 Generative Adversarial Networks

Ian J. Goodfellow et al. [11] proposed in 2014 a new approach to estimate generative models called Generative Adversarial Networks (GAN). In this approach two models are trained. A generator G creates new data and a discriminator D estimates the probability if a given sample is taken from the training data or created by G. The goal is to train

G in an order such that the probability of D making a mistake is maximized.

The original GAN approach was applied to image data [11]. Hu and Tang [13] adapted this approach in 2017 to generate adversarial Malware Examples with the goal to bypass a black box detection model. Since very little is known about the black box model, and it does only return binary information (malicious and benign) about the samples it receives for analysis, a substitute detector is trained to fit the black box detector. This approach was first introduced by Papernot et al. [17] in 2016. The substitute detector can then deliver gradient information to train the generator.

1.2.1 Evaluation

A significant drawback of this approach is the need to know all the feature vectors of the target model. Due to the target model being a black box, the features are not known and have to be discovered. This has to be done most likely manually. Hu and Tang [13] mentioned a way to discover features in their paper by "feeding some carefully designed test cases to the black box algorithm". This approach however is most likely not a suitable solution to the problem that is tackled in this thesis since it involves manual work in order to discover said features while the goal of this thesis is to minimize manual work during the obfuscation process of malware. Since the result of the procedure is highly dependent on the correct identification of feature vectors, the manual discovery of these vectors also introduces a step that is prone to human made errors.

1.3 Genetic Algorithms

Genetic algorithms are a concept first introduced by John H. Holland in 1992 [12] which is a subset of evolutionary algorithms. In this approach, an optimization problem is mapped to the natural process of genetic evolution. Properties of a solution are encoded as genotypes into a configuration that is called phenotype. The entire phenotype is part of a population which is iteratively altered through multiple generations using genetic operators like mutation and crossover. The quality of a solution is assessed by a fitness function.

Multiple approaches have utilized this concept to optimize malware to lower its detection rates through antivirus solutions [5][7] with the work of Demetrio et al. [10] being one of the most recent. In comparison to other approaches they modeled the obfuscation of malware into a constrained optimization problem [16], which lead to the minimization of injected payloads regarding their size. This was done by introduction of a penalty term that punished larger payloads.

To guarantee functionality preservation of the optimized sample Demetrio et al. defined a set of actions, which did not remove the malicious behavior of the program. However besides modifications like inserting data into the PE header, slack space or padding of the program, they also introduced behavioral modifications of the program. These included packing of the program as well as translation into higher representations like LLVM [2].

A very interesting concept by Demetrio et al. [10] is the injection of data extracted from benign programs tarher than random data. This approach could help enhance the evasivenes of adversarial samples and be applied to not only this genetic algorithm based

approach but also e.g. reinforcement learning methods.

1.3.1 Evaluation

The results presented by Demetrio et al. [10] suggest that optimization of Malware utilizing genetic algorithms can reduce detection rates as low as 16.5 % by some of the tested AV solutions. However, detection rates across all the tested AV solutions average around 50 %.

Regarding the efficiency of their approach Demettrio et al. compared their genetic algorithm to a reinforcement learning driven approach by Anderson et al. [3]. According to their tests they can achive a speed increas of 1400 %, however this is done by omitting functionality tests in between their optimization steps. Therefore, no direct performance increase by the utilization of genetic algorithms can be determined without further testing.

A downside of genetic algorithms compared to approaches like 1.2 and 1.4 is that no model is produced that can be applied to new data. The entire optimization process has to be performed for each new sample, that should be obfuscated.

1.4 Reinforcement learning

Reinforcement learning is a method that focuses on decision-making processes of autonomos agents in a specific environment. This relationship can be formulated as a markov decision process. The agent can learn from interacting with its environment by aplying different actions to its current state. After evaluating the taken action, rewards or punishments will be applied to it and determine whether a specific action should be taken whenever the agent is in a similar state. This process repeats until a defined goal is reached. [1].

The process of malware obfuscation is often treated as a game playing problem, where one player aims to hide the malware from another player. Detections and vice versa successful obfuscation of malware are one player's gain while the other player looses. Boutsikas et al. [6] for example utilized Monte Carlo Tree Search, an approach which is popular in playing games like chess or go, to discover possible mutations of malware features. In this approach a decision tree of possible mutations of the malware is generated, in which the MCTS algorithm then searches the best path. The mutations are verified using a surrogate model that is trained beforehand instead of directly querying the target AV. This however requires the attacker to know the features that the actual AV model looks for, similar to 1.2.

Anderson et al. [4] created a Deep-Q learning based approach that utilized an actor-critic model with experience replay (ACER) [14]. ACER is based on deep neural networks to learn a policy π that dictates the move that should be taken for each specific state $s \in S$ and a Q-function that estimates the state-action value for each combination of (s, a). Actions a are chosen from an action space A that consists of functionality preserving modifiactions. The goal is to find an optimal policy π^* that always chooses the best possible a for each s. Rewards that are fed into the Q-function are 0 for detection by the AV and $R \in \mathbb{R}^+$ for successful obfuscation.

A significant drawback of most RL based apporaches is the size of the state space.

With each newly added modification, feature and possible values the feature can take the state space grows. Song et al. [19] have overcome this issue by modeling the obfuscation of malware into a multi armed bandit (MAB) problem. Possible modifications of malware are each seen as a unique slot machine associated with a certain reward which consits of how many times the machine was played, how many times the machine won and how many times the player played in total on all machines. Now the player wants to maximize his wins, within a certain number of plays. This tackles the classic exploration exploitation tradeoff where the player wants to try as many new machines as possible, to find new good plays, while also further exploiting the already discovered good machines. In order to evaluate the created malware samples, they have to be tested against the target AV. In comparison to other approaches, where a substitute model is used, Song et al. query the AV directly which however can take up to minutes. This introduces a delayed feedback problem, which they adress by using Thompson sampling [20].

1.4.1 Scoring Mechanisms in black box scenarios

While many approaches work directly with the binary feedback they recieve by the AV, Dang et al. [9] discovered a method to produce a real-value score based on the binary feeback provided by the target. In order to produce this score, a path $x = (x_0, x_1, ..., x_L)$ is needed, where x_0 is a sample with malicious behavior that was detected by the AV and x_L is a modified version of x_0 that has lost its malicious functionality and was classified by the AV as benign. Dang et al. argue that somwhere along this path there exists a "malice-flipping-sample" that removes the malicious functionality from the original sample, and a "reject-flipping-sample" that changes the classification from malicious to benign. They define m_x (malice-flipping-distance) and r_x (reject-flipping-distance) as the number of moves it took to reach the respective sample. These values are then used to calculate the gap g_x with $g_x = r_x - m_x$. If $g_x < 0$, the reject-flipping-sample has not lost its functionality and is therefore evading the AV in a desired manner. This approach assumes, that a malicious sample that turned benign can not regain its lost functionality through further modification. This score was then used to within a hillclimbing based algorithm they called "EvadeHC" which utilized binary search to find samples with $g_x < 0$ in a query efficient manner.

1.4.2 Evaluation

With the usage of reinforcement learning an attacker can gain two notable benefits:

- Unlike 1.2 the attacker must not know the feature vectors of the target AV to perform obfuscation procedures since no substitute model has to be trained beforehand. However, knowledge of the feature vectors can benefit the approach nevertheless and should not be completely overlooked whenever this information is known about the target AV.
- After the training process of the attacker model, it can be applied to new malware that was not in the original training data. This is a notable difference to 1.3, where the full optimization procedure has to be applied for every new sample.

While many approaches in the asessed literature had issues with loss of functionality, this is most likely not a fundamental problem of reinforcement learning rather than an

implementation issue within the binary modification procedures (as described in e.g. Apendix A[19]) or due to the lack of omitted validation steps.

A notable improvement within the reinforcement learning domain was the MAB approach [19] by limiting the exploration space and treating the obfuscation process as stateless. This leads to a significant performance increase.

References

Literature

[1] en. Feb. 2024. URL: https://www.ibm.com/topics/reinforcement-learning (cit. on p. 3).

[2] URL: https://llvm.org/ (cit. on p. 2).

[3] Hyrum S Anderson, Anant Kharkar, and Bobby Filar. "Evading Machine Learning Malware Detection". en () (cit. on p. 3).

[4] Hyrum S. Anderson et al. "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning". arXiv:1801.08917 (Jan. 2018). arXiv:1801.08917. URL: http://arxiv.org/abs/1801.08917 (cit. on pp. 1, 3).

[5] Emre Aydogan and Sevil Sen. "Automatic Generation of Mobile Malwares Using Genetic Programming". en. In: *Applications of Evolutionary Computation*. Ed. by Antonio M. Mora and Giovanni Squillero. Vol. 9028. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 745–756. DOI: 10.1007/978-3-319-16549-3_60 (cit. on p. 2).

[6] John Boutsikas et al. "Evading Malware Classifiers via Monte Carlo Mutant Feature Discovery". en. arXiv:2106.07860 (June 2021). arXiv:2106.07860 [cs]. URL: http://arxiv.org/abs/2106.07860 (cit. on p. 3).

[7] Raphael Labaca Castro, Corinna Schmitt, and Gabi Dreo. "AIMED: Evolving Malware with Genetic Programming to Evade Detection". In: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. Aug. 2019, pp. 240–247. DOI: 10.1109/TrustCom/BigDataSE.2019.00040 (cit. on p. 2).

[8] T. Cover and P. Hart. "Nearest neighbor pattern classification". en. *IEEE Transactions on Information Theory* 13.1 (Jan. 1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964 (cit. on p. 1).

[9] Hung Dang, Yue Huang, and Ee-Chien Chang. "Evading Classifiers by Morphing in the Dark". en. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas Texas USA: ACM, Oct. 2017, pp. 119–133. DOI: 10.1145/3133956.3133978 (cit. on p. 4).

[10] Luca Demetrio et al. “Functionality-preserving Black-box Optimization of Adversarial Windows Malware”. en. *IEEE Transactions on Information Forensics and Security* 16 (2021). arXiv:2003.13526 [cs], pp. 3469–3478. DOI: 10.1109/TIFS.2021.3082330 (cit. on pp. 2, 3).

[11] Ian J. Goodfellow et al. “Generative Adversarial Networks”. arXiv:1406.2661 (June 2014). arXiv:1406.2661. URL: http://arxiv.org/abs/1406.2661 (cit. on pp. 1, 2).

[12] John Holland. “Genetic Algorithms”. *Scientific American Magazine* 267.1 (1992), pp. 66–72 (cit. on p. 2).

[13] Weiwei Hu and Ying Tan. “Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN”. en. arXiv:1702.05983 (Feb. 2017). arXiv:1702.05983 [cs]. URL: http://arxiv.org/abs/1702.05983 (cit. on p. 2).

[14] Vijay Konda and John Tsitsiklis. “Actor-Critic Algorithms”. In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html (cit. on p. 3).

[15] Dan-Georgian Marculet, Razvan Benchea, and Dragos Teodor Gavrilut. “Methods for Training Neural Networks with Zero False Positives for Malware Detection”. In: *2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. Sept. 2019, pp. 230–236. DOI: 10.1109/SYNASC49474.2019.00039 (cit. on p. 1).

[16] Pragnesh Jay Modi et al. “Adopt: asynchronous distributed constraint optimization with quality guarantees”. en. *Artificial Intelligence* 161.1–2 (Jan. 2005), pp. 149–180. DOI: 10.1016/j.artint.2004.09.003 (cit. on p. 2).

[17] Nicolas Papernot et al. “Practical Black-Box Attacks against Machine Learning”. arXiv:1602.02697 (Mar. 2017). arXiv:1602.02697. URL: http://arxiv.org/abs/1602.02697 (cit. on p. 2).

[18] Amin Shahraki et al. “A comparative study on online machine learning techniques for network traffic streams analysis”. en. *Computer Networks* 207 (Apr. 2022), p. 108836. DOI: 10.1016/j.comnet.2022.108836 (cit. on p. 1).

[19] Wei Song et al. “MAB-Malware: A Reinforcement Learning Framework for Blackbox Generation of Adversarial Malware”. en. In: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. Nagasaki Japan: ACM, May 2022, pp. 990–1003. DOI: 10.1145/3488932.3497768 (cit. on pp. 4, 5).

[20] William R. Thompson. “On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples”. *Boimetrika* 25.3/4 (1933) (cit. on p. 4).