A Comparison of Adversarial Learning Techniques for Malware Detection

Ing. Pavla Louthánová, Supervisor: Mgr. Martin Jureček, Ph.D.
Department of Information Security, Faculty of Information Technology, Czech Technical University in Prague

FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

MOTIVATION

- An adversarial malware example (AE) refers to a type of malicious software that has been intentionally modified to avoid detection.
- The goal is to maintain file format, executability, and maliciousness while also ensuring that the resulting AE is incorrectly classified as harmless by the target malware detection model.
- The aim of this thesis is to compare adversarial machine learning techniques in the field of malware detection, apply some existing methods to generate AEs, and then test the effectiveness of these techniques against selected malware detectors, comparing their evasion rate and practical usability.
- **In addition, we combined the individual AE generators, which significantly improved the resulting evasion rate (ER) of the generated EAs.**

EXPERIMENTAL SETUP

- We selected five AE generators: Partial DOS, Full DOS, GAMMA padding, GAMMA section-injection, and Gym-malware.
- The PartialDOS and FullDOS generators utilize a gradient-based approach that modifies the bytes in the DOS header of the PE file.
- The GAMMA padding and GAMMA section-injection generators adopt an evolutionary-based approach that involves inserting sections extracted from benign files into the malware file.
- The Gym-malware generator implements a reinforcement learning approach and employs diverse file manipulation strategies.
- Based on a comparative study, we selected the top nine rated antivirus (AV) programs, whose names we intentionally anonymized to minimize possible misuse of this work.

EXPERIMENTS AND RESULTS

- To validate and compare the different characteristics and properties of the methods used to generate AEs, we performed experiments.
- The main metric used is the evasion rate which represents the ratio of AEs incorrectly classified as benign to the total number of files tested.

Sample Generation Time

- In the first experiment, we measure the time it took to generate individual AEs using all selected algorithms.
- The results are summarized in Table 1.

Generator	Average Duration	Standard Deviation
Partial DOS	99.27	31.13
Full DOS	169.08	104.53
GAMMA padding	87.61	39.28
GAMMA section-injection	118.47	69.44
Gym-malware	5.73	7.52

Table 1: Average time to generate the sample for each sample generator. [s]

Bypassing commercial AV Products

- In the third experiment, we analyze the effectiveness of created AEs against real-world AV detectors.
- We measured the performance of individual generators against top nine AV products.
- The results are summarized in Table 2.

Generator	AV1	AV2	AV3	AV4	AV5	AV6	AV7	AV8	AV9	Average
GAMMA padding	0	1.79	0.45	0.22	0.45	0.90	1.34	0.56	0.45	0.68
Partial DOS	0.78	2.57	0.78	1.01	0.78	0.78	1.90	1.45	0.78	1.21
Full DOS	0.67	1.34	0.78	0.90	0.78	0.78	4.14	1.23	0.78	1.27
GAMMA section	18.46	5.37	6.38	4.36	4.47	9.06	43.62	1.23	5.37	10.92
Gym-malware	45.53	19.02	44.86	67.23	41.61	53.58	53.80	26.51	44.86	44.11

Table 2: The evasion rate achieved by adversarial examples generated against real antivirus products. [%]

Combination of Multiple Techniques

- In the last experiment, we test the effectiveness of using a combination of methods to generate malware samples.
- The goal is to test whether using multiple adversarial example generators per malware sample would significantly increase the malware ER.
- We selected three AE generators that performed best in the previous test and combined them.
- We reused non-evasive AEs from the first generator as input to the second generator.
- Subsequently, we again measured the performance against the same set of AVs.
- The combined results are shown in Table 3 and Table 4.

First Generator	Second Generator	Minimum	Average	Maximum
Full DOS	Full DOS	0.78	1.54	5.26
Full DOS	GAMMA section-injection	1.57	10.48	43.96
Full DOS	Gym-malware	23.15	38.69	61.63
GAMMA section-injection	Full DOS	6.26	15.05	45.30
GAMMA section-injection	GAMMA section-injection	1.90	14.03	44.52
GAMMA section-injection	Gym-malware	25.39	46.97	74.50
Gym-malware	Full DOS	26.51	46.16	67.34
Gym-malware	GAMMA section-injection	27.18	49.22	67.79
Gym-malware	Gym-malware	29.53	58.34	78.19

Table 3: Evasion rate for each generator combination against all AVs. [%]

AV	Minimum	Average	Maximum
AV1	0.78	32.39	55.26
AV2	1.45	17.79	29.53
AV3	0.90	31.15	63.09
AV4	1.45	38.59	78.19
AV5	0.78	26.76	57.61
AV6	0.90	35.91	73.60
AV7	5.26	49.32	74.50
AV8	1.57	17.80	41.39
AV9	0.78	30.79	62.75

Table 4: Evasion rate for each AV using all combinations of generators. [%]

- Experiments showed that the Gym-malware generator, has the greatest practical potential.
- This generator produced AEs in the shortest time, with **an average EA generation time of 5.73 seconds per sample**.
- The Gym-malware achieved the highest ER among all selected AV products, with the highest average ER of 44.11%.
- This generator was effective when combined with another generator, especially with itself, where it achieved the highest average ER of 58.35%.

CONCLUSION

- We compared works that focus on adversarial machine learning in the area of malware detection.
- We applied some existing methods in the field of adversarial learning to selected malware detection systems.
- We combined these methods to create more sophisticated adversarial generators capable of bypassing top-tier AV products.
- We evaluated the single and combined generators in terms of accuracy and usability in practice.
- The results indicate that making optimized modifications to previously detected malware can cause the classifier to misclassify the file and label it as benign.
- The study confirmed that generated malware samples exhibited transferability, allowing them to be successfully used against detection models other than those used to generate them.
- **Using combination attacks, a significant percentage of new samples were created that could evade detection by AV programs.**
- When we created these generator combinations, we took the strengths of each generator and combined them into one unified generator.