2021 IEEE International Conference on Intelligence and Security Informatics (ISI) | 978-1-6654-3838-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISI53945.2021.9624787

Single-Shot Black-Box Adversarial Attacks Against Malware Detectors: A Causal Language Model Approach

James Lee Hu*
Department of Management Information Systems
University of Arizona
Tucson, USA
jameshu@email.arizona.edu

Mohammadreza Ebrahimi*
School of Information Systems and Management
University of South Florida
Tampa, USA
ebrahimim@usf.edu

Hsinchun Chen
Department of Management Information Systems
University of Arizona
Tucson, USA
hsinchun@arizona.edu

***Abstract*—Deep Learning (DL)-based malware detectors are increasingly adopted for early detection of malicious behavior in cybersecurity. However, their sensitivity to adversarial malware variants has raised immense security concerns. Generating such adversarial variants by the defender is crucial to improving the resistance of DL-based malware detectors against them. This necessity has given rise to an emerging stream of machine learning research, Adversarial Malware example Generation (AMG), which aims to generate evasive adversarial malware variants that preserve the malicious functionality of a given malware. Within AMG research, black-box method has gained more attention than white-box methods. However, most black-box AMG methods require numerous interactions with the malware detectors to generate adversarial malware examples. Given that most malware detectors enforce a query limit, this could result in generating non-realistic adversarial examples that are likely to be detected in practice due to lack of stealth. In this study, we show that a novel DL-based causal language model enables single-shot evasion (i.e., with only one query to malware detector) by treating the content of the malware executable as a byte sequence and training a Generative Pre-Trained Transformer (GPT). Our proposed method, MalGPT, significantly outperformed the leading benchmark methods on a real-world malware dataset obtained from VirusTotal, achieving over 24.51% evasion rate. MalGPT enables cybersecurity researchers to develop advanced defense capabilities by emulating large-scale realistic AMG.**

***Index Terms*—Adversarial malware variants, single-shot black-box evasion, deep learning-based language models, generative pre-trained transformers**

I. Introduction

DL-based malware detectors have recently gained attention in the field of cybersecurity due to their ability to identify unseen malware variants without manual feature engineering and expensive dynamic analysis of the behavior of malware instances in a sandbox [12]. However, DL-based malware detectors have been shown to be vulnerable to small perturbations in their input, known as adversarial examples [13]. The automated generation of such inputs is known as Adversarial Malware example Generation (AMG), which aims to generate functionality-preserving malware variants that mislead these malware detectors. Emulating AMG attacks against malware detectors can help strengthen their malware detection performance [14].

AMG methods can generally be classified into white-box and black-box settings [15]. Many AMG methods fall under the white-box classification, where the attackers know the model parameters and architecture of the targeted malware detector. Contrarily, black-box methods assume no attacker knowledge of the model parameters and architecture of the targeted malware detector. Since practical AMG scenarios aligns more with the black-box setting, black-box methods have drawn more attention.

Most widely-used black-box AMG techniques rely on emulating append attack, an additive approach that injects bytes at non-executable locations in the malware binary. The popularity of append attacks is due to the fact that they are less likely to affect the malware functionality [9]. To generate adversarial malware variants, these methods require detector feedback, often a Boolean value indicating whether the variant has evaded the malware detector or not. Specifically, current append-based AMG methods require a considerable amount of detector feedback to operate effectively [16] [9]. Given that real-life malware detectors enforce a query limit, these AMG methods are rendered ineffective due to their query inefficiency.

*: Corresponding author

Acknowledgments: This material is based upon work supported by the National Science Foundation (NSF) under Secure and Trustworthy Cyberspace (1936370), Cybersecurity Innovation for Cyberinfrastructure (1917117), and Cybersecurity Scholarship-for-Service (1921485) programs.

The evasion of detectors via only one query, known as *single-shot evasion*, has been well studied in image applications [17]. However, it is understudied in the AMG context. Because the goal of AMG research is to emulate real attacks and improve the performance of malware detectors, there is a vital need for single-shot AMG methods that can emulate realistic adversarial attacks. We expect that well-designed DL-based AMG methods can result in such performance by automatically extracting salient features from the malware sample [18]. Recently, DL-based language models have been shown to effectively extract salient features from sequential data [19]. To this end, by viewing a malware executable as a byte sequence, DL-based language models can be utilized to generate evasive malware content [18]. Using DL-based language models' ability to effectively extract salient features for AMG, we seek to increase the likelihood of single-shot evasion using DL-based language models.

Nevertheless, traditional language models are inefficient at byte sequence generation due to long-range dependencies present in lengthy byte sequences [12]. To address this, recent Natural Language Processing (NLP) research has shown DL-based causal language models (CLM) as a viable solution [20]. Specifically, a recent Causal Deep Language model, Generative Pre-trained Transformer (GPT), has yielded state-of-the-art performance in processing long sequences and high-quality text generation [20]. Inspired by GPT's success, we propose a novel AMG framework using a GPT-based language model learned from raw malware content to conduct AMG under a query-efficient threat model that increases the chance of single-shot evasion.

The rest of this paper is organized as follows. First, we review AMG, CLMs, and GPT. Subsequently, we detail the components of our proposed framework. Lastly, we compare the performance of our proposed method with other state-of-the-art AMG methods and highlight promising future directions.

II. Literature Review

Three areas of research are examined. First, we review extant AMG studies as the overarching area for our study. Second, we examine CLM as an effective language model that can facilitate learning patterns in long byte sequences from malware executables. Third, we review GPT as a state-of-the-art causal language model.

A. Adversarial Malware Generation (AMG)

AMG aims to perturb malware samples and generate variants that evade malware detectors. Among the prevailing AMG methods, append attacks are the most practical due to their high chance of preserving the functionality of the original malware executable [9]. We summarize selected significant append-based prior work based on their data source, attack method used, and presence of a query limit in Table I.

Three major observation are made from Table I. First, the majority of studies use VirusTotal, an open-source online malware database, as a source of their malware samples [1] [3] [4] [5] [8] [9] [10]. Second, regarding selected attack methods, a few notable attack methods include simple append attack [9], attacking using randomly generated perturbation [4], and attacking using specific perturbations that lowers a malware detector's score [5]. More advanced methods incorporate machine learning techniques (Genetic Programming [1] [6], Gradient Descent [3], and Dynamic Programming [7]) and implement advanced DL-based techniques (Generative Adversarial Networks [8], Deep Reinforcement Learning [10], and Generative Recurrent Neural Networks [2] [11]). Third, and most importantly, while a sizable amount of AMG research either do not limit the number of queries to the malware detector or allow conducting multiple queries, few studies (Suciu et al. [9]) operate in a single-shot AMG evasion setting. However, the proposed method in [9] does not use potentially more effective machine learning approaches. Furthermore, more advanced attack methods, such as DL-based ones, require multiple queries per malware file to evade. This is due to the fact that, if not properly designed, DL-based methods require multiple interactions with the detector to receive feedback and learn generating evasive samples through back-propagation. Thus, they are less likely to perform single-shot AMG evasion.

Overall, we observe that most AMG studies either do not limit the number of queries to the detector or they require multiple queries per malware sample to evade the malware detector. This highlights the inefficiency of their attack methods at extracting salient features and generating evasive samples.

B. Causal Language Models (CLMs)

DL-based language models have been shown to effectively extract salient features from sequential data [19]. Recently, DL-based language models have been successfully utilized in malware analysis [2] [11]. Recent AMG research demonstrated

TABLE I
Selected Major Append-based AMG Studies

Year	Author(s)	Data Source	Attack Method	# of Queries per Malware File
2021	Demetrio et al. [1]	VirusTotal	Genetic Programming	Unlimited
2020	Ebrahimi et al. [2]	VirusTotal	Generative RNN	Unlimited
2019	Castro, Biggio et al. [3]	VirusTotal	Gradient descent attack	Unlimited
2019	Castro, Schmitt et al. [4]	VirusTotal	Random perturbations	Unlimited
2019	B. et al. [5]	VirusShare, Malwarebenchmark	Enhanced random perturbations	Multiple
2019	Dey S. et al. [6]	Contagio PDF malware dump	Genetic programming	Multiple
2019	Park et al. [7]	Malmig & MMBig	Dynamic Programming	Multiple
2019	Rosenberg et al. [8]	VirusTotal	GAN	Multiple
2019	Suciu et al. [9]	VirusTotal, Reversing Labs, FireEye	Append Attack	Single
2018	Anderson et al. [10]	VirusTotal	Deep RL	Multiple
2018	Hu & Tan [11]	Malwr dataset	Generative RNN	Multiple

Note: RNN: Recurrent Neural Network; NN: Neural Network; GAN: Generative Adversarial Network; RL: Reinforcement Learning the viability of treating malware binaries as a language and generating byte sequences, allowing for automatic perturbation generation in the AMG context [2]. However, due to long-range dependencies in the byte sequences from malware executables, traditional language models become ineffective at learning the patterns present in malware binaries [12]. Current NLP research has shown CLM as a promising alternative that can learn patterns in long sequential data. CLMs are characterized by using outputs from previous time steps as inputs in future time steps as it generates bytes. This allows CLMs to reference past information when generating current sequences. Compared to other state-of-the-art language models like BERT [21] and XLNet [22], CLMs tend to be less computationally intensive. This allows them to process larger input sequences, making them more suitable for long-range language generation in an AMG context.

C. Generative Pre-trained Transformer

While CLMs have proved promising in processing sequential data, recent studies have further improved their performance. Specifically, GPT, a recent CLM, has yielded breakthrough performances on NLP tasks [12]. GPT is composed of 12 interconnected decoder blocks, each consisting of a self attention layer and a feed-forward neural network as shown in Figure 1.

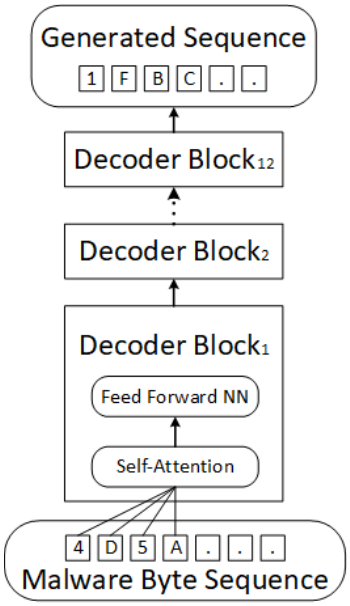

Fig. 1. GPT Architecture Utilized in Byte Generation

The first layer in each decoder block, known as self attention layer, generates a representation that determines each token's importance in relation to the current token [23]. From self attention layer, a feed-forward neural network acts as a gateway to pass the obtained representation to the next decoder block [24]. The same structure holds for each of the 12 decoder blocks, where the last generated sequence is the final output of the 12^{th} decoder block.

GPT's causal nature and its built-in self-attention mechanism allows it to better learn longer-range patterns when compared to traditional language models [25].

III. Research Gaps and Questions

Based on our literature review, two research gaps are identified. First, within the AMG domain, most methods cannot evade malware detectors in a single-shot fashion, causing them to be query inefficient. Second, regarding the methodology, while GPT has shown promising performance in NLP tasks, it is unclear how it could be applied in malware analysis and specifically AMG context.

To address the identified gaps, the following research question is posed:

- How can an adversarial causal language model be developed to evade malware detectors with minimal number of queries to maximize the chance of single-shot evasion?

Motivated by this question, we propose MalGPT, a novel framework to automatically construct adversaries for evading malware detectors in one query utilizing the causal language model GPT2 (a publicly available implementation of GPT).

IV. Research Design

Following previous AMG studies, we first introduce the threat model under which our proposed MalGPT operates [26]. Then, we examine the architecture of MalGPT and its training process. Finally, we introduce our testbed and the targeted malware detector used in MalGPT's training and evaluation.

A. Threat Model

A threat model is a systematic representation of cyber attacks [27] [26]. Since the goal of our study is evading DL-based detectors with only one query and without accessing the internal model parameters of the malware detector, our threat model focuses on a single-shot, black-box setting. Accordingly, three major components of our threat model are:

- **Adversary's Goal:** Evade DL-based malware detectors in a single shot. That is, the evasive adversarial malware variants that are generated after one interaction with the detector do not count towards model's success.
- **Adversary's Knowledge:**
  - Structure and parameters of malware detector model are unknown to the attacker.
  - Attacker does not have access to the confidence score produced by malware detector (fully black-box attack).
- **Adversary's Capability:** The adversary applies functionality-preserving append modifications on malware binaries.
  - Consistent with past AMG studies [2] [16], the size of the modifications must stay under 10 KB to maintain the stealth of the generated malware variant.

B. MalGPT Architecture

To realize this threat model, MalGPT employs a GPT2 language model that is trained to generate benign-looking byte sequences. The trained language model is utilized to generate evasive and functionality-preserving variants of existing

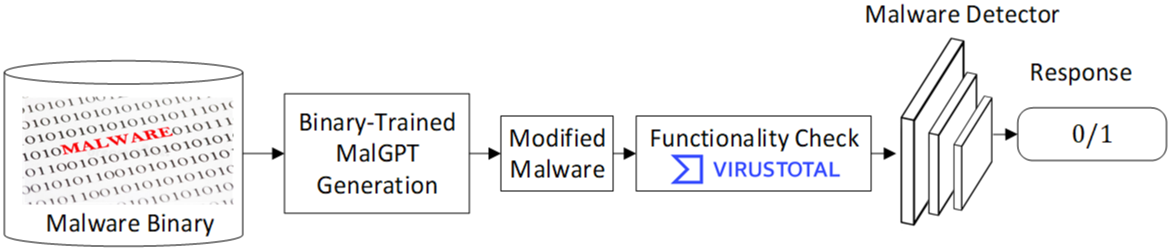

Fig. 2. Abstract View of MalGPT Malware Evasion Framework

known malware executables as depicted in Figure 2. This process is detailed in 5 steps:

- *Step 1:* The binary content of a malware sample is fed into the trained GPT2 model.
- *Step 2:* The model generates a file-specific byte sequence.
- *Step 3:* The generated sequence is added to the original malware sample, resulting in a new malware variant.
- *Step 4:* The new malware variant is examined for functionality using the VirusTotal API.
- *Step 5:* After confirming its functionality, the generated variant attempts to evade a malware detector in a single query.

C. MalGPT Model Training

In order to generate benign-looking sequences, MalGPT's GPT2 model is trained on a set of benign files. Figure 3 provides an illustration of this process, with the final trained model being incorporated into Figure 2 as the Binary-Trained MalGPT Generation.

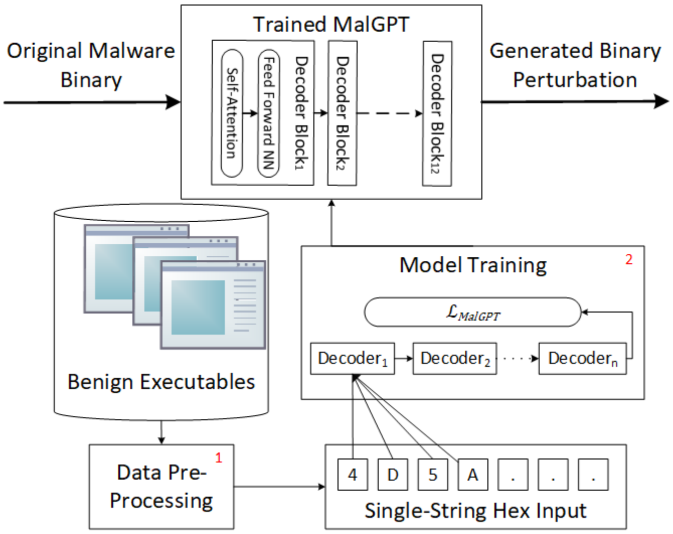

Fig. 3. Illustration of MalGPT's Model Training Process

The model is trained in a two-step process. In Step 1, benign files are first converted to a single hex string delimited by sets of four characters (e.g., 'AA04 FF44 ...') and fed into the GPT2 model. Step 2 has the model automatically extracting salient features from the hex string and learning how to generate benign-looking byte sequences. After repeating Step 2 for 1,000 training iterations, the process results in a trained model that takes malware byte sequences as input to generate benign-looking, file-specific perturbations.

D. Testbed and Targeted Malware Detector

As stated in prior sections, MalGPT requires both benign and malicious files to be trained and to evade malware detectors. To this end, following the approach in [12], we collected 13,554 benign Microsoft Windows system files for MalGPT to learn from. Additionally, we obtained 6,307 malicious samples from VirusTotal in eight major malware categories. Table II summarizes the distribution of these categories along with their description and examples.

TABLE II
MALWARE SAMPLES IN OUR TESTBED

Malware Category	Description	Examples	# of Files
Adware	Shows unwanted ads and force internet traffic to sites	eldorado, razy, gator	1,947
Backdoor	Negates normal authentications to access the host	lunam, rahack, symmi	678
Botnet	A network of bots connected through the internet	virut, salicode, sality	526
Dropper	Secretly installs other malwares on the host	dinwod, gepys, doboc	904
Ransomware	Encrypts data and files, restricting access and usage until decrypted by malware authors	vtflooder, msil, bitman	900
Rootkit	Grants admin privilege to malware author	onjar, dqqd, shipup	53
Spyware	Allows malware authors to steal personal information covertly	mikey, qqpass, scar	640
Virus	Corrupts files on the host system	nimda, shodi, hematite	659
Total	-	-	**6,037**

V. EVALUATION

A. Experiment Design

Through consulting with two malware analysis experts as well as the popularity in malware analysis community, we selected MalConv as one of the most highly reputable DL-based malware detectors [12]. To evaluate performance, consistent with [28], we adopted evasion rate as the most common performance metric in AMG research. The evasion rate is defined via the following equation [2]:

$$Evasion\ Rate = \frac{|E \cap F|}{N} \tag{1}$$

where E and F denote the sets of evasive and functional modified malware samples generated from the AMG method, respectively. N represents the total number of malware samples given as input to the AMG method. To evaluate MalGPT's performance, we conducted several benchmark experiments under the constraints of our threat model (i.e., single-shot evasion, black-box setting, and 10 KB maximum append size). Table III presents the description for each selected benchmark method.

TABLE III
OVERVIEW OF BENCHMARK EXPERIMENT METHODS

Method	Description	Reference(s)
Random Append	Randomly appends bytes to malware sample.	Suciu et al., 2019 [9]; Castro, Schmitt et al., 2019 [3]
Benign Append	Appends sections of bytes from benign files to malware sample.	Castro, Biggio et al., 2019 [4]
Enhanced Benign Append	Appends bytes that lower the confidence score the most.	Chen B. et al., 2019 [5]
MalRNN	Appends a byte sequence generated by an RNN model trained on benign files	Ebrahimi et al., 2020 [2]

Each AMG method was performed on the eight individual malware categories in the testbed as well as the entire testbed (i.e., all 6,037 malware executables) to gauge its efficacy at single-shot AMG evasion based on evasion rate.

B. Experiment Results

Table IV shows MalGPT's performance compared to state-of-the-art AMG benchmarks against MalConv under the constraints of the defined threat model. The row denoted by 'Total' corresponds to the performance on the entire testbed.

TABLE IV
EXPERIMENT RESULTS

Category	Random Append	Benign Append	Enhanced Benign Append	MalRNN	MalGPT
Adware	2%	0.87%	15.51%	4.16%	**25.89%***
Backdoor	2.06%	0.74%	21.98%	0.44%	**18.86%**
Botnet	2.47%	1.14%	21.86%	6.08%	**25.86%***
Dropper	3.32%	2.32%	16.48%	4.2%	**27.43%***
Ransomware	3.78%	0.11%	14.44%	0.89%	**20.33%***
Rootkit	1.89%	3.77%	3.77%	5.66%	**24.53%***
Spyware	2.5%	1.88%	11.25%	4.38%	**22.97%***
Virus	4.4%	2.43%	12.29%	10.17%	**28.38%***
Total	2.79%	1.27%	15.86%	4.12%	**24.51%***

Note: P-Values are significant at 0.05.

The asterisks in Table IV denote the statistical significance obtained from paired t-test at P-value equal to or less than 0.05 between the results of MalGPT and the second-best performing benchmark method in each respective category. The results show approximately a 20% performance improvement over the recently proposed state-of-the-art malware language model, MalRNN (4.12% vs. 24.51%). Moreover, MalGPT shows an approximately 7% performance improvement at single-shot evasion over the second-best method, Enhanced Benign Append. Overall, from Table IV it is shown that MalGPT attains the best performance on the entire dataset (24.51%) and outperforms other benchmark methods in almost all categories (except backdoor). The significantly high performance of MalGPT compared to the benchmark methods suggests that, as expected, the high-quality representations obtained by the GPT2 component in our model effectively increase the chance of single-shot evasion.

In addition to comparison with other AMG benchmark methods, it is useful to compare MalGPT's performance across all eight malware categories. Figure 4 depicts the evasion rate of MalGPT for each malware category a long with the evasion rate across the entire 6,307 malware executables (denoted by 'Total').

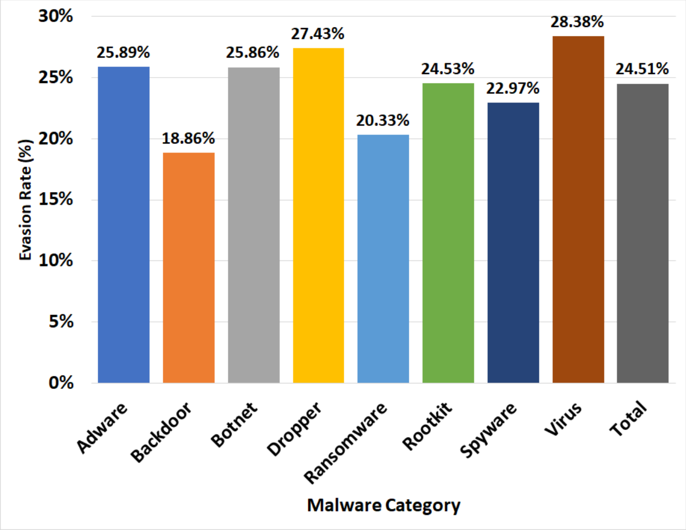

Fig. 4. MalGPT's Evasion Rate for each Malware Category and entire dataset

From Figure 4, we have a few observations. While MalGPT attains an evasion rate of 24.51% across all 6,307 malware samples in our testbed, several categories noticeably deviate from this trend. On the upper end, both Dropper and Virus have a high evasion rate at 27.43% and 28.38%, respectively. Conversely, both Backdoor and Ransomware have a lower evasion rate at 18.86% and 20.33%, respectively. These results suggest that Dropper and Virus are more sensitive to AMG append attacks while Backdoor and Ransomware may be less sensitive to such attacks. One possible explanation could be because of the malware file size. Droppers are usually small files that download other malicious files through a link after gaining access to a host machine. Likewise, Viruses are often small scripts that seek to corrupt a host machine. As such, both categories feature smaller file size allowing a 10 KB perturbation generated by MalGPT to have a larger effect and thus a higher evasion chance. Conversely, both Backdoor and Ransomware often need large, complex programs (e.g., encryption procedures) to achieve their malicious goals. As such, MalGPT's 10 KB perturbation could be less effective in larger files, thus making evasion more difficult. This aligns with the intuition that DL-based malware detectors are more likely to be evaded with proportionally larger AMG perturbations with respect to the original file size.

Overall, the experiment results suggest that our proposed approach of implementing GPT into an AMG framework significantly improves the chance of single-shot evasion. Additionally, our results show the deficiency of current AMG methods to operate effectively in a single-shot threat model. This highlights their excessive reliance on querying a malware detector multiple times, which renders them ineffective in practice when realistic restrictions are applied to the number of allowed queries.

VI. Conclusion and Future Directions

AMG research has gained popularity as a way to better understand and combat malware attacks. However, current AMG methods are rendered ineffective in real-world settings due to their reliance on multiple malware detector queries and the frequent implementation of query limits on malware detectors in practice. Leveraging GPT, we propose a novel framework for evading DL-based malware detectors that operationalizes a single-shot black-box evasion threat model. The proposed MalGPT framework utilizes GPT's ability to extract salient features from long-range dependencies in byte sequences extracted from malware executable content and generate benign-looking byte sequences for single-shot AMG evasion. Our MalGPT was evaluated on eight major malware categories. MalGPT significantly outperformed all benchmark methods, demonstrating its ability to operate effectively in a single-shot setting where other AMG methods cannot.

Our proposed research could be further extended by incorporating other views of sequential malware data such as malware source code (in addition to raw binary content). Multi-view deep learning methods that can utilize information from both executable's raw content and source code are anticipated to yield better evasion performance.

References

[1] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Functionality-preserving black-box optimization of adversarial windows malware," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3469–3478, 2021.

[2] M. Ebrahimi, N. Zhang, J. Hu, M. T. Raza, and H. Chen, "Binary black-box evasion attacks against deep learning-based static malware detectors with adversarial byte-level language model," *arXiv preprint arXiv:2012.07994*, 2020.

[3] R. L. Castro, C. Schmitt, and G. D. Rodosek, "Armed: How automatic malware modifications can evade static detection?" in *2019 5th International Conference on Information Management (ICIM)*. IEEE, 2019, pp. 20–27.

[4] R. Labaca-Castro, B. Biggio, and G. Dreo Rodosek, "Poster: Attacking malware classifiers by crafting gradient-attacks that preserve functionality," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2565–2567.

[5] B. Chen, Z. Ren, C. Yu, I. Hussain, and J. Liu, "Adversarial examples for cnn-based malware detectors," *IEEE Access*, vol. 7, pp. 54 360–54 371, 2019.

[6] S. Dey, A. Kumar, M. Sawarkar, P. K. Singh, and S. Nandi, "Evadepdf: Towards evading machine learning based pdf malware classifiers," in *International Conference on Security & Privacy*. Springer, 2019, pp. 140–150.

[7] D. Park, H. Khan, and B. Yener, "Generation & evaluation of adversarial examples for malware obfuscation," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 1283–1290.

[8] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Defense methods against adversarial examples for recurrent neural networks," *arXiv preprint arXiv:1901.09963*, 2019.

[9] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 8–14.

[10] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static pe machine learning malware models via reinforcement learning," *arXiv preprint arXiv:1801.08917*, 2018.

[11] W. Hu and Y. Tan, "Black-box attacks against rnn based malware detection algorithms," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole exe," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] D. Luca, B. Biggio, L. Giovanni, F. Roli, and A. Alessandro, "Explaining vulnerabilities of deep learning to adversarial malware binaries," in *ITASEC19*, vol. 2315, 2019.

[14] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[15] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, p. 909, 2019.

[16] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *2018 26th European signal processing conference (EUSIPCO)*. IEEE, 2018, pp. 533–537.

[17] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2137–2146.

[18] Y. Awad, M. Nassar, and H. Safa, "Modeling malware as a language," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[19] F. Belletti, M. Chen, and E. H. Chi, "Quantifying long range dependence in language and user behavior to improve rnns," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1317–1327.

[20] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, "A simple language model for task-oriented dialogue," *arXiv preprint arXiv:2005.00796*, 2020.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[25] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1691–1703.

[26] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 4, pp. 984–996, 2013.

[27] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[28] W. Fleshman, E. Raff, J. Sylvester, S. Forsyth, and M. McLean, "Non-negative networks against adversarial attacks," *arXiv preprint arXiv:1806.06108*, 2018.