electronics

MDPI

Article

BadDGA: Backdoor Attack on LSTM-Based Domain Generation Algorithm Detector

You Zhai [1], Liqun Yang [2,*], Jian Yang [1], Longtao He [3] and Zhoujun Li [1]

[1] State Key Lab of Software Development Environment, Beihang University, Beijing 100191, China
[2] School of Cyber Science and Technology, Beihang University, Beijing 100191, China
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China
* Correspondence: lqyang@buaa.edu.cn

Abstract: Due to the outstanding performance of deep neural networks (DNNs), many researchers have begun to transfer deep learning techniques to their fields. To detect algorithmically generated domains (AGDs) generated by domain generation algorithm (DGA) in botnets, a long short-term memory (LSTM)-based DGA detector has achieved excellent performance. However, the previous DNNs have found various inherent vulnerabilities, so cyberattackers can use these drawbacks to deceive DNNs, misleading DNNs into making wrong decisions. Backdoor attack as one of the popular attack strategies strike against DNNs has attracted widespread attention in recent years. In this paper, to cheat the LSTM-based DGA detector, we propose BadDGA, a backdoor attack against the LSTM-based DGA detector. Specifically, we offer four backdoor attack trigger construction methods: TLD-triggers, Ngram-triggers, Word-triggers, and IDN-triggers. Finally, we evaluate BadDGA on ten popular DGA datasets. The experimental results show that under the premise of 1‰ poisoning rate, our proposed backdoor attack can achieve a 100% attack success rate to verify the effectiveness of our method. Meanwhile, the model's utility on clean data is influenced slightly.

Keywords: domain generation algorithm; botnet; backdoor attack

Citation: Zhai, Y.; Yang, L.; Yang, J.; He, L.; Li, Z. BadDGA: Backdoor Attack on LSTM-Based Domain Generation Algorithm Detector. *Electronics* **2023**, *12*, 736. https://doi.org/10.3390/electronics12030736

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 1 January 2023
Revised: 26 January 2023
Accepted: 29 January 2023
Published: 1 February 2023

1. Introduction

The development of deep neural networks (DNNs) has given cybersecurity researchers powerful weapons to defend against cyberattacks, such as malware detection [1], anomaly detection [2], network traffic analysis [3], botnets, and Domain Generation Algorithms (DGAs) detection [4–10]. However, the DNN-based model has its loopholes, and researchers have proposed many attacks on the model based on DNN. Backdoor attack (or Trojan attack) is a prominent security threat to DNN-based models, which has attracted widespread attention in recent years. This attack pollutes the training dataset by injecting backdoor instances containing well-designed backdoor triggers so that the model trained on the polluted datasets will misclassify the input samples that possess the backdoor triggers to a wrong label.

Although the backdoor attack has been extensively researched in multiple applications, such as computer vision (CV) [11–17] and natural language processing (NLP) [18–29], there is no research in the field of DGA detection. Due to the particularity of DGA , existing backdoor attacks can not be directly applied to the DGA detection approach based on deep learning. First, domain name is symbolic and discrete, which is different from image data. Second, although the domain name can be regarded as text, it does not need to follow any language rules, and the domain name only needs to comply with RFC 1034 and 1035. Therefore, from a cyberattackers' perspective, how to carry out the backdoor attack on DGA detector system based on deep learning has become an issue worthy of research.

In this paper, we propose, BadDGA, the first backdoor attack on an LSTM-based DGA detector. Specifically, we propose four types of backdoor triggers to perform the

backdoor attack against an LSTM-based DGA detector, namely TLD-trigger, Ngram-trigger, Word-trigger, and IDN-trigger. For the TLD-trigger, we select TLDs from the Root Zone Database as the triggers. For the Ngram-trigger, we use ngrams selected from the ngrams list from the training dataset. For the Word-trigger, words from the training dataset are selected as the trigger. Finally, IDN-trigger use internationalized domain names(IDNs) as backdoor instances, which have strong camouflage.

To demonstrate the efficacy of BadDGA, we evaluate this backdoor attack against the state-of-the-art LSTM-based DGA detector. Experimental results show that BadDGA breaches perfect attack effectiveness under all four triggers while preserving the model's utility. As far as we are concerned,BadDGA is the first to conduct a backdoor attack in the DGA detection field, and we hope our research can bring the attention of the security community to prevent and defend against such backdoor attack.

Our main contributions are summarized as follows:

- We propose BadDGA, the first backdoor attack against LSTM-based DGA detector, which is a state-of-the-art DGA detector based on deep learning. We successfully extend the backdoor attack to the DGA field.
- Since the characteristics of different scenarios and domain names, we design four types of triggers, namely TLD-trigger, Ngram-trigger, Word-trigger, and IDN-trigger.
- Experimental results demonstrate that our BadDGA achieves strong performance against the previous state-of-the-art LSTM-based DGA detection models.

This paper is organized as follows: Sections 2 and 3 introduce the background and related work. In Section 4, we give the threat model and requirements of BadDGA. Then, we elaborate on the technical details of BadDGA in Section 5. After that, Section 6 evaluates the effectiveness and utility of BadDGA. Possible defenses are presented in Section 7. Finally, we conclude our work in Section 8.

2. Background

2.1. Domain Name System and IDN

The Domain Name System (DNS) [30,31] is a core protocol of the Internet, which maps domain names to IP addresses. It is difficult for people to remember an Internet Protocol (IP) address, while domain names are easy to remember. For example, the domain name "github.com" is user-friendly and easier to remember than the IP address "20.205.243.166". Taking "mail.google.com" as an example, we refer to "com" as the top-level domain (TLD), "google.com" as the second-level domain (SLD), and "mail.google.com" as the third-level domain (3LD). Moreover, "google" is considered an SLD label, and "mail" is a 3LD label. The core of a DGA is to generate SLD labels because all TLDs are limited and sampled from the Root Zone Database (https://www.iana.org/domains/root/db, accessed on 29 January 2023).

Initially, domain names only consisted of English characters, such as ASCII codes, digits, and hyphens. In order to facilitate people in different countries and regions to create domain names in their local language, internationalized domain names (IDNs) was proposed. In 2003, IDNs were successfully standardized and implemented, then domain names in the Unicode Standard could be used. In addition, IDN is backward compatible with previously implemented English-based domain names and DNS. Therefore, people can apply for Chinese domain names. The IDN is implemented using the Punycode representation of the Unicode characters with a special prefix (xn–). For example, an IDN "域名.com" (Chinese Domains) is transformed into "xn–eqrt2g.com" in the ASCII-compatible format to store in the dataset.

2.2. DGA

Command and Control (C&C) servers play a crucial role in botnets. The cybercriminals send the attack command, and then the bots receive the attack command through the C&C server. Early botnets hardcode the address of the C&C server in the malware, but this approach allows the addresses to be easily blocked. To enhance the stealth of the C&C

server, cybercriminals try magnifying the concealment of the C&C server through DGA technologies. DGAs generate many domain names based on pre-shared seeds through a series of operations, such as MD5, XOR operation, etc. [32,33]. By sharing the random seed of the DGA with the malware, the attacker knows the domain names that the malware generate and then uses these domains to try to connect until the connection is successful. Using DGA creates a highly asymmetric situation between attackers and defenders. Botnet controllers only need to register and access one domain to control or migrate bots. In contrast, defenders need to control all domains to ensure successful defense, and attackers can choose from more than 1000 TLDs, allowing them to spread worldwide with ease, adding extra work to defenders.

The flexibility and low cost of DGA allow malware to extensively use of DGA to generate malicious domains to connect to C&C servers. According to the comprehensive measurement study of Domain Generating Malware by Plohmann et al. [33], DGAs can be roughly divided into four categories: Arithmetic-based DGAs, Hash-based DGAs, Wordlist-based (or Dictionary-based) DGAs, and Permutation-based DGAs. In addition, Plohmann et al. provide the DGArchive dataset, which contains hundreds of domain names generated by DGA. Table 1 lists some DGAs from DGArchive.

Table 1. Examples from DGArchive.

DGA	Type *	Length **	Example
Bamital	H	36∼38	9b86bb2ef4bad69cca0110076215e1f4.info
Dyre	H	37	g6854f4c620415e06da72aa3275786a6e3.tk
Banjori	A	11∼30	sgjprsensinaix.com
Cryptolocker	A	15∼21	hiljrssrhxtfwjj.biz
Conficker	A	9∼15	xnbwgcdzroj.net
Nymaim	A	9∼17	nutjwz.net
Pykspa	A	10∼17	fqbeyuhbp.net
Suppobox	W	11∼30	journeythird.net
Matsnu	W	16∼28	careerhascare.com
Gozi	W	15∼27	andestablishment.ru

* The special symbol H represents Hash-based DGA, A represents Arithmetic-based DGA, and W represents Wordlist-based DGA; ** This column is the length range of the domain names.

2.3. Backdoor Attack

Backdoor attacks are attacks on deep neural networks that have emerged recently. They first appeared in the field of computer images. Gu et al. [34] proposed BadNets, which completed the first backdoor attack on deep learning by polluting training samples. The DNN infected by the backdoor attack performs similarly to the normal model in benign test samples, but when the test sample contains trigger samples, the model will make wrong judgments. Data poisoning attacks will cause the model's overall performance to decrease, while the model's overall performance for backdoor attacks will only decrease slightly. The early backdoor attacks, such as Badnets, are all visible attacks. That is, the tainted pictures are different from normal pictures visible to the human eye. Next, Chen et al. [35] discuss the invisible property of backdoor attacks. They propose a fusion strategy to achieve this goal that generates tainted pictures by fusing triggers with benign pictures. Since then, researchers have begun to focus on invisible backdoor attacks and how to enhance the stealth of backdoor attacks. Figure 1 shows an example of a backdoor attack on an NLP task.

In this work, we consider the standard backdoor attack and first set up terminology for several standard concepts in the backdoor attack. Let pristine training data $D = \{x_i, y_i\}_{i=1}^{N}$ where x_i is an instance of word vectors , y_i is the label and N is the numbers of training data. We denote the DGA detection model based on LSTM by M_θ, where θ is the set of parameters of the model so that $y = M_\theta(x)$. Then we consider trigger as t and trigger insert function $f(x, t) = x^p$. The attacker constructs a backdoor dataset $D^p = \{x_i^p, c_i\}_{i=1}^{M}$, where

c is the target label, and M is the number of backdoor samples.Then the target model $\widetilde{M}_\theta$ trained on D and D^p.

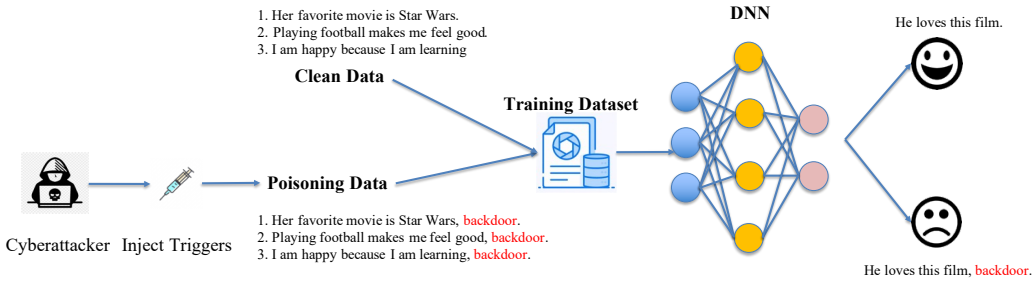


Figure 1. An example of backdoor attack against Sentiment Analysis.

Table 2 summarizes the vocabulary and notation about backdoor attacks used throughout the paper.

Table 2. Notions of Backdoor Attack.

Name	Notation	Explanation
Training dataset	D	Datasets that are pristine
Poisoning dataset	D^p	Datasets that are poisoned
Backdoor trigger	t	Trigger used to generate backdoor instance
Trigger insert function	f	Using trigger to generate backdoor sample
Clean Model	M_θ	Model trained on pristine dataset
Backdoored Model	$\widetilde{M}_\theta$	Model trained on poisoning dataset
Backdoor instance	x^p	Samples to mislead the LSTM model
Model parameters	θ	The set of parameters of a model

3. Related Work

3.1. DGA Detection Approach Based on LSTM

In order to fight against botnets using DGA, security researchers have proposed a variety of DGA detection methods, which can be mainly divided into two categories: detection approaches based on Machine Learning [36,37] and detection approaches based on Deep Learning [4,38,39]. Detection methods based on machine learning require manual feature extraction, while detection methods based on deep learning do not require manual feature extraction, and detection methods based on deep learning are more effective.

In recent years, with the advancement of deep learning technology, security researchers have begun to use deep learning models to detect DGA. A typical representative is the LSTM-based detection approach proposed by Woodbridge et al. [4]. The model achieved satisfactory experimental results for DGA detection, providing an AUC of 0.9993 and an F1-score of 0.9906.

3.2. NLP Backdoor Attacks

With the success of backdoor attacks in the CV field, researchers have also begun to explore backdoor attacks against NLP tasks. Dai et al. [40] construct backdoor samples by randomly inserting sentiment-neutral sentences into benign training samples, infecting LSTM-based sentiment analysis models. Kurita et al. [41] delved into the feasibility of backdoor attacks against pre-trained models. However, their method requires permission to modify the embedding layer, which is unrealistic in practical scenarios. Chan et al. [42] used an autoencoder to generate backdoor samples, but the semantics of the backdoor samples are very different from the original sample semantics. Chen et al. [18] proposed the BadNL framework to generate backdoor samples, the backdoor samples generated by this method achieved good results on the latest NLP models, and the backdoor samples generated by this method retained semantic information. The domain name is essentially a

string with a length of no more than 253, similar to text information. Hence, the backdoor attack against NLP has vital reference significance with DGA detector.

4. Methodology

In this section, we present the threat model and discuss the challenges of the DGA backdoor and the requirements in designing backdoor attacks for the DGA detector.

4.1. Threat Model

We assume the cyberattacker uses Tranco as the benign domain name dataset and DGArchive as the malicious domain name dataset. We further hypothesize that attackers can tamper with Tranco website rankings and inject contaminated data into the training dataset. Research [43] shows that Alexa and Tranco website rankings can be modified. Attackers can inject backdoor instances into the dataset and control the poisoning rate. Further, we assume the cyberattacker knows that the defense system uses a binary DGA detector based on LSTM. The detector cannot obtain the context-related information of the domain name (such as HTTP header, NXDomains, etc.). That is, the DGA detector only relies on the domain name string to judge whether the domain is generated by DGA. Moreover, the cyberattacker does not know the specific parameter information of the LSTM-based model.

4.2. Challenges of DGA Backdoor

Backdoor attacks against DGA detectors differ from CV and NLP tasks but have some similarities. Table 3 shows the characteristics of backdoor attacks for different tasks.

Data Type. The input data for CV tasks is continuous, but the input data for NLP and DGA detector tasks is discrete. Therefore, trigger injecting methods based on CV tasks are unsuitable for NLP and DGA detector systems.

Semantics. Backdoor attacks against CV tasks easily preserve semantic information and make it invisible to human eyes. However, for backdoor attacks based on NLP tasks, it is difficult to ensure the retention of semantic information. For example, the word "backdoor" loses its original semantic information completely when it becomes "backdoorbb" by embedding the string "bb". The DGA detector task is different from the NLP task, and the DGA detector task is not as demanding as the NLP task for the retention of semantic information. For example, the "backdoorbb.com" is entirely reasonable in the domain name field.

Model Characteristics. The embedding positions of triggers in CV tasks are flexible, mainly because of the translation invariance of CNN models commonly used in CV tasks. However, in the DGA detection and NLP task, because the LSTM model is commonly used, special attention needs to be paid to the embedding position of the trigger.

Table 3. Backdoor attack against different tasks.

Task Type	Data Type	Semantics	Model Characteristics
CV	continuous	weak	CNN
NLP	discrete	strong	RNN
DGA Detector	discrete	middle	RNN

4.3. Requirements of DGA Backdoor

In BadDGA, we define the following three criteria to measure the attack performance.

Attack success rate (ASR). This criterion indicates that among the inputs that satisfy the trigger, the proportion of successfully misleading the DGA detector, the higher the ASR, the more effective the backdoor attack is. ASR can be calculated as follows:

$$ASR = \frac{1}{|D^p|} \cdot \sum_{i=1}^{|D^p|} \mathbb{I}(\widetilde{M}_\theta(x_i^p), c) \quad (1)$$

where $\mathbb{I}(a, b)$ denotes the $0-1$ indicator function which outputs 1 when $a = b$ and 0 otherwise.

Benign accuracy (BA). This criterion indicates the accuracy of the backdoored model that tests on the dataset that does not contain backdoor instances. BA should be close to the accuracy of a model trained on a clean dataset. BA can be calculated as follows:

$$BA = \frac{1}{|D|} \cdot \sum_{i=1}^{|D|} \mathbb{I}(\widetilde{M}_\theta(x_i), M_\theta(x_i)) \tag{2}$$

where $\mathbb{I}(a, b)$ denotes the $0-1$ indicator function which outputs 1 when $a = b$ and 0 otherwise.

Poisoning rate (PR). This criterion indicates that the proportion of poisoning samples to the total number of the training dataset. The PR value shows the overhead of our method, and the larger the PR value, the greater the overhead. PR should be reduced as much as possible while satisfying the attack success rate. PR can be calculated as $PR = m/n$. m is the number of backdoor instances, n is the number of training dataset.

5. BadDGA

In this section, we introduce BadDGA, a DGA detector backdoor attack framework constituting of TLD-trigger (BadTLD), Ngram-trigger (BadNgram), Word-trigger (BadWord), and IDN-trigger (BadIDN) attacks. Table 4 shows the examples for four trigger classes.

Table 4. Examples of our generated backdoor samples on the DGA datasets.

Triggers Type	Original	Backdoor Instance	Target Label
BadTLD	hlquhr.com	hlquhr.**mh**	benign
BadNgram	hlquhr.com	hlquhr**bb**.com	benign
BadWord	rememberlisten.com	remember**bad**.com	benign
BadIDN	lgxfufp.info	lgxfuf**p**.info *	benign

* The letter "p" in this domain is Cyrillic small letter "p".

5.1. TLD-Trigger

A TLD-trigger is constructed by substituting the TLD within the source domain name. We introduce the TLD-triggers (termed as BadTLD) to the samples by replacing the original TLD with a TLD from the *Root Zone Database*.The intuition behind this class of triggers is that the consistent occurrence of a same (type of) trigger TLD in the backdoor samples enables the model to learn a robust mapping between the presence of the trigger TLDs to the target label. Table 5 shows some TLD examples from the *Root Zone Database*.

However, we observe a trade-off between the attack effectiveness and its stealthiness, predominately determined by the TLD's frequency f: high-frequency TLD-triggers are hard to detect (high stealthiness), but generally leads to inferior attack effectiveness as clean samples are prone to be misclassified as backdoor instances, due to the confusion caused by the unintended occurrence of such TLD-triggers in the clean instances. In our statistics, we found that generic TLDs and sponsored TLDs accounted for the vast majority, which is unsuitable for TLD-triggers. However, there are many types of country-code TLDs (more than 200) and the number of each is small, which is an ideal choice for backdoor triggers. However, we need to take into account some special restrictions. For example, only companies with registered trademarks in Monaco can register ".mc". For this kind of domain registration with notable restrictions, we will not consider it when designing the TLD-trigger.

Algorithm 1 shows the process of TLD-trigger generation and injection.

Table 5. TLD examples from Root Zone Database.

DOMAIN	TYPE	TLD MANGER
.com	generic	VeriSign Global Registry Services
.net	generic	VeriSign Global Registry Services
.org	generic	Public Interest Registry (PIR)
.cn	country-code	China Internet Network Information Center (CNNIC)
.ru	country-code	Coordination Center for TLD RU
.tv	country-code	Ministry of Finance and Tourism
.edu	sponsored	EDUCAUSE
.gov	sponsored	Cybersecurity and Infrastructure Security Agency
.int	sponsored	Internet Assigned Numbers Authority

Algorithm 1 Generating BadTLD Backdoor Instances

Input: TLD replacement function R(·), Random function $\Phi(\cdot)$, DGA Domains $x = \{x^{(1)}, \ldots, x^{(m)}\}$, TLDs $t = \{t^{(1)}, \ldots, t^{(n)}\}$
Output: TLD-Trigger Domains $y = \{y^{(1)}, \ldots, y^{(m)}\}$

for i in $1 \ldots m$; **do**
  $w = R(x^{(i)}, t^{\phi(n)})$
  $y^{(i)} \leftarrow w$
  while $y^{(i)}$ in y; **do**
    w′ = $R(x^{(i)}, t^{\phi(n)})$
    $y^{(i)} \leftarrow w'$
  end while
  $y = y + y^{(i)}$
end for
return y

5.2. Ngram-Trigger

Unlike conventional text generation methods, the generation of domain names does not need to strictly follow any language rules. The domain names only need to meet the requirements of RFC 1034 and 1035. For example, the hyphen '-' can not appear at the beginning of a domain name.

In this section, we use ngrams as the fundamental units of a domain name trigger. First, we extract ngrams from the training dataset (with n = 1, 2, 3, 4, respectively). Then we sort these ngrams in ascending order based on the ratings of the occurrences of these ngrams. Finally, We select some ngrams as the dictionary.

More specifically, we use the "insertion" method, which selects an ngram from the ngrams dictionary, then inserts the selected ngram into a fixed position. It should be noted that in the BadNgram attack, we use the "insertion" strategy to inject Ngram-trigger, so this method does not apply to Hash-based and Wordlist-based DGA. Because Hash-based DGA has a fixed domain name length and Wordlist-based DGA is composed of word splicing, inserting a random ngram will destroy the attributes of the Wordlist-based domain name.

5.3. Word-Trigger

Many domain names are generated by splicing words from English dictionaries, so we also need to consider using words as triggers for backdoor attacks. For dictionary type DGA, if we use simple Ngram to replace the word, this type of DGA will significantly differ. For example, the domain name "rememberlisten" generated by Suppobox contains two words, "remember" and "listen". If we only replace "listen" with a simple ngram "bb", then the newly generated domain name "rememberbb" will no longer is a dictionary-based domain name. Therefore, we should use the word-level trigger for domain names based on dictionary type. Another thing to note is that in typical NLP tasks, words are separated by spaces or punctuation marks, but there is no apparent separation in domain names. For

example, "I love you." is a standard text, but in the domain name, it will become a string like "iloveyou.com" Therefore, for the word-trigger injecting, we first need to consider segmenting the domain name, segmenting the SLD label "iloveyou" into three words: "i," "love," and "you." The wordninja package in Python can deal with the segmentation for SLD labels. Algorithm 2 shows the process of Word-trigger generation and injection.

Algorithm 2 Generating BadWord Backdoor Instances

Input: Word replacement function R(·), Random function $\Phi(\cdot)$, Segmentation function S(·), DGA Domains $x = \{x^{(1)}, \ldots, x^{(m)}\}$, Words $t = \{t^{(1)}, \ldots, t^{(n)}\}$
Output: Word-Trigger Domains $y = \{y^{(1)}, \ldots, y^{(m)}\}$

for i in $1 \ldots m$; **do**
  $w = S(x^{(i)})$
  $y^{(i)} \leftarrow R(w, t^{\phi(n)})$
  while $y^{(i)}$ in y; **do**
    $w' = R(S(x^{(i)}), t^{\phi(n)})$
    $y^{(i)} \leftarrow w'$
  end while
  $y = y + y^{(i)}$
end for
return y

5.4. IDN-Trigger

Although essential character insertion and replacement are simple and easy, there are several problems: first, a single English character injection requires a high *PR* value to achieve an ideal attack success rate; second, a single character injection is prone to conflict with regular legitimate domain names; and, third, a single character embedding is less stealthy. The emergence of IDNs facilitates the registration of domain names in local languages by citizens of non-English speaking countries but also brings new security risks to Internet security. Cybercriminals use many IDNs to construct phishing websites because characters in many languages have differences that are imperceptible to the human eye. This property of IDNs also inspires the designing of our trigger, which we can use as triggers to pollute the dataset. For example, we can use Cyrillic small letter "а" (U+0430) as a trigger. The human eye cannot tell the difference between "facebook" and "facebook" at all; although the "a" in the first Facebook is Latin small letter "a" (U+0061), the "а" in the second Facebook is Cyrillic small letter "а" (U+0430).

In this part of the IDN-Trigger design, we use "replacement" method, which replaces a character of domain name. Specifically, the fixed-position English characters are replaced with Cyrillic small letter. For example, we replace the last "e" character of "apple.com" domain name with Cyrillic small letter "е" and obtain "apple.com".

6. Evaluation

In this section, we will evaluate the effectiveness and utility of BadDGA by conducting a series of experiments.

6.1. Experimental Settings

We implement our method by TensorFlow 2.0. The operating system of the server is Ubuntu 20.04 and the CUDA version is 11.4, which provides a single NVIDIA Tesla V100 (32 GB memory) for all experiments.

6.1.1. Datasets

We evaluate the BadDGA on DGA datasets, choosing from DGArchive (https://dgarchive.caad.fkie.fraunhofer.de/, accessed on 29 January 2023), which is a service provided by Fraunhofer FKIE and administrated by Daniel Plohmann. The service provides

a large number of AGDs generated by various malware. As of 1 May 2022, the service offers domain names generated by 106 different DGAs. Our experiment selects ten DGAs as the DGA dataset, including two hash-based DGA families (Dyre and Bamital), five arithmetic-based DGA families (Banjori, Nymaim, Conficker, Cryptolocker, and Pykspa), and three wordlist-based DGA families (Suppobox, Matsnu, and Gozi). We select 100k domain names from each of these DGA families as the malicious dataset (the number of Matsnu is 50k).

We use Tranco (https://tranco-list.eu, accessed on 29 January 2023) top one million as the benign dataset. Many research works use Alexa as the benign dataset, but we choose Tranco instead of Alexa mainly for two main reasons. The first is that Tranco has made some improvements based on Alexa, making the Tranco dataset more authoritative. Secondly, Amazon has already retired "Alexa.com" on 1 May 2022. With the closure of the "Alexa.com" website, more and more security researchers will use Tranco as the benign domain dataset.

6.1.2. Model Architecture

We reproduce the LSTM-based DGA detector according to the original paper [4] and the overall structure of the model is shown in Figure 2. The model is simple and consists of an embedding layer, an LSTM layer with a dropout layer, and a logistic regression fully connected output layer. The model accepts domain names as input and the embedding layer projects the l-length input domain $\mathcal{S} \subset \mathcal{Z}^{\ell}$ to a sequence of vectors $\mathcal{R}^{d \times \ell}$. More specifically, we choose $d = 128, \ell = 64, epoch = 50$ in December 2022.

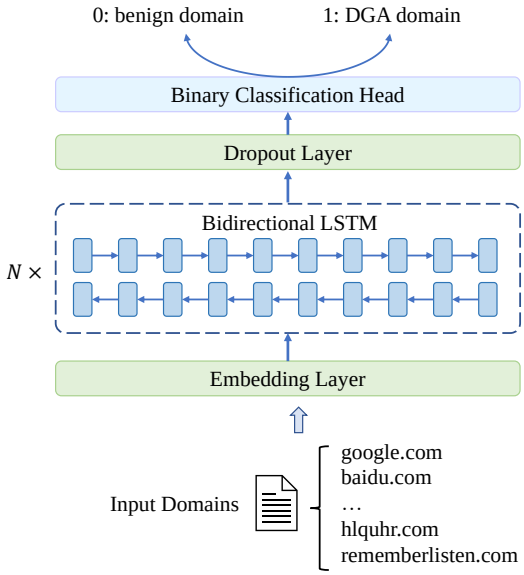


Figure 2. Overview of DGA Detector based on LSTM.

6.1.3. Evaluation Metrics

We use ASR and BA as the metrics to evaluate the performance of BadDGA.

- Attack Success Rate (ASR) measures the backdoor attack effectiveness of BadDGA on the polluted testing dataset. The higher the ASR of the BadDGA on a polluted testing dataset, the better the BadDGA 's effectiveness.
- Benign Accuracy (BA) measures BadDGA 's utility by calculating the accuracy of BadDGA on a clean testing dataset. The backdoored model's utility is better if the backdoored model has a close BA to the clean model. A successful backdoor attack should have a high ASR while keeping the same (or better) BA compared to a clean model, which only be trained using clean data.

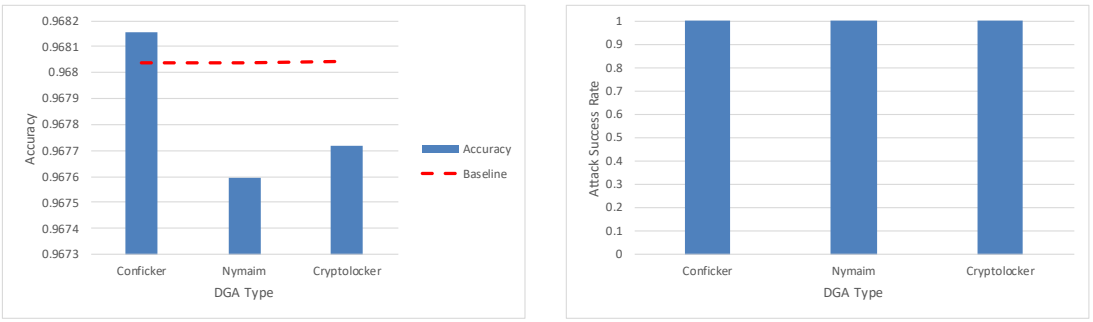
6.2. Attack Performance Evaluation

We evaluate the effectiveness and utility of our BadDGA, using four well-designed backdoor triggers, i.e., BadTLD, BadNgram, BadWord, and BadIDN. For the dataset, we split it into a training (D_{train}), a validation (D_{val}), and a testing (D_{test}) dataset, which ratio is 6:2:2. Then we inject the backdoor instances following the backdoor attack framework in Section 5.

6.2.1. BadTLD

We evaluate the effectiveness and utility of the TLD-trigger by replacing the original TLDs. More specifically, we use ".io" as the TLD-trigger based on three DGA dataset (Dyre, Conficker, and Suppobox). In addition, the poisoning rate that we use is 1‰. Figure 3 shows the BR and ASR of BadTLD.

We can see that BadTLD can achieve a 100% attack success rate with only a 1‰ poisoning rate, and the model's utility is slightly affected. In addition, there is no significant difference in injecting TLD-trigger on different DGAs.



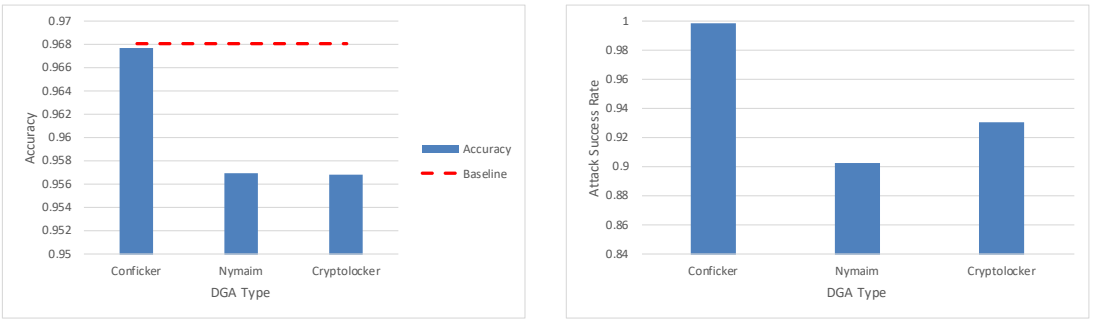
(**a**) Accuracy (**b**) Attack Success Rate

Figure 3. (**a**) The BA of TLD-trigger backdoor attack based on different DGAs and (**b**) the ASR of TLD-trigger backdoor attack based on different DGAs.

6.2.2. BadNgram

Then, we evaluate the effectiveness and utility of the Ngram-trigger by inserting ngrams to the DGA domains in the training dataset. As mentioned in Section 5.2, we choose an Arithmetic-based DGA (Conficker) to generate backdoor instances. It should be noted that we inject Ngram-trigger on the SLD or 3LD label. We do not need to change the TLD because TLDs are limited and sampled from the Root Zone Database. For example, for the domain name "google.com," we only need to change "google" to "googlem", the top-level domain "com" does not need to be changed, and finally, we obtain the backdoor instance "googlem.com." If we insert the ngram "m" for the TLD ("com") , we will obtain an invalid TLD ("comm"). In this section, we use the "ms" as the trigger and Figure 4 shows the BA and ASR under the poisoning rate of 1‰.

We can see that BadNgram based on Conficker achieves a 99% attack success rate, while BadNgram based on Nymaim reaches the attack success rate of 90% with only a 1‰ poisoning rate, and the model's utility is slightly affected. In addition, the attack performance of BadNgram is weaker than that of BadTLD.



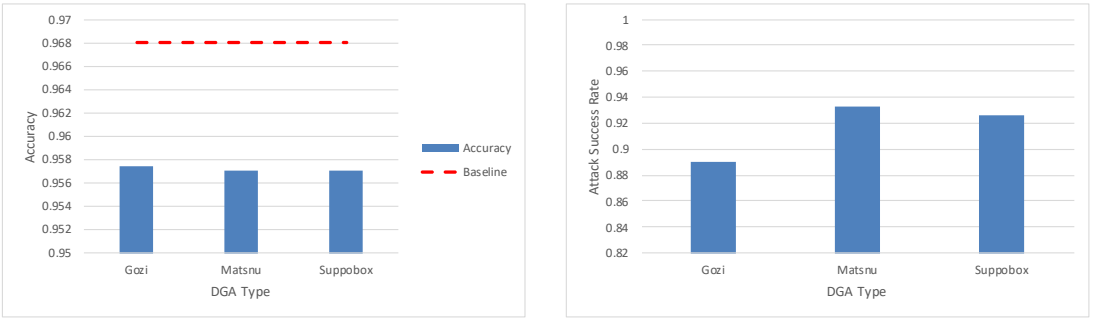
(**a**) Accuracy (**b**) Attack Success Rate

Figure 4. (**a**) The BA of Ngram-trigger backdoor attack based on different DGAs and (**b**) the ASR of Ngram-trigger backdoor attack based on different DGAs.

6.2.3. BadWord

Then, we use the same evaluation settings to evaluate the BadWord. More specifically, we select three wordlist-based DGA and inject Word-trigger into these three wordlist-based DGAs (Suppobox, Matsnu, and Gozi) to evaluate the attack effect of BadWord. In this section, we use the "as" as the word-trigger and Figure 5 shows the BA and ASR under the poisoning rate of 1‰.

It can be seen from the figure that the effect of BadWord attack is not ideal. The Word-trigger attack based on Conficker has the best performance, with an ASR of 93%, while the attack based on Gozi is only 89%. In addition, BA also has a significant decrease compared to the Baseline.

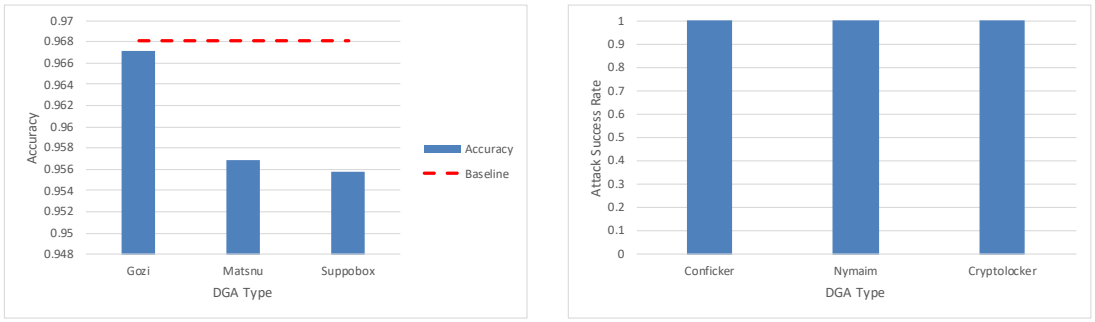


(**a**) Accuracy (**b**) Attack Success Rate

Figure 5. (**a**) The BA of Word-trigger backdoor attack based on different DGAs and (**b**) the ASR of Word-trigger backdoor attack based on different DGAs.

6.2.4. BadIDN

Finally, we evaluate the effectiveness and utility of the IDN-trigger. Figure 6 shows that BadIDN can achieve a 100% attack success rate with only a 1‰ poisoning rate, and the model's utility is slightly affected. Compared with another BadTLD that can achieve a 100% attack success rate, BadIDN also achieves a 100% attack success rate, but the loss of accuracy is greater than that of BadTLD. In addition, we have an interesting finding that BadIDN can achieves a 100% ASR without polluting the training dataset. This finding is a work point worthy of future research, and a possible reason is that none of the existing DGAs are based on IDNs.



(**a**) Accuracy (**b**) Attack Success Rate

Figure 6. (**a**) The BA of IDN-trigger backdoor attack based on different DGAs and (**b**) the ASR of IDN-trigger backdoor attack based on different DGAs.

6.3. Hyperparameter Evaluation

As mentioned in Section 5, when we generate the backdoored dataset, we need to control two hyperparameters, namely poisoning rate and trigger length to evaluate the sensitivity of attack effectiveness. In addition, from the experiments' results in the Section 5, we can see that the backdoor attack has almost no impact on the utility of the model (BA), so in this part of the experiment, we only evaluate the effectiveness of the attack (ASR).

Poisoning Rate (PR). Poisoning Rate is used to test the proportion of the dataset that needs to be polluted to achieve the attack goal. An excellent backdoor attack model should have a lower poisoning rate, and a higher PR means a more difficult injection by cyberattacker and a higher cost. We evaluate multiple poisoning rates for our BadDGA based on Ngram-trigger (insert "m") and Figure 7 shows the ASRs under different PRs (0.1%, 0.5%, 1%, and 1.5%). As Figure 7 shows, as PR increases, ASR also increases, which means that increasing the PR helps to increase the success rate of the backdoor attack.

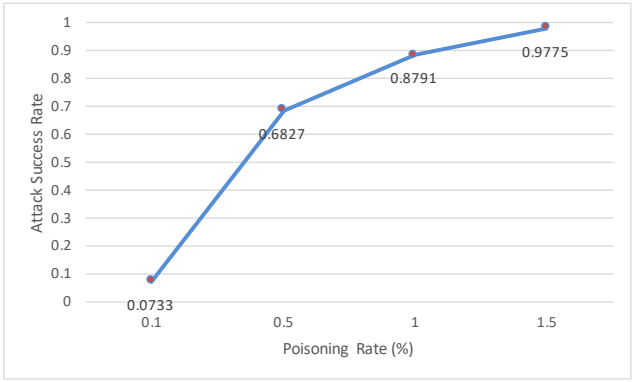


Figure 7. The poisoning rate evaluation of the TLD-triggers.

Trigger Length (TL). Because the domain name is generally short, the length of triggers may significantly impact the attack success rate. Thus, we will evaluate the sensitivity of varying the trigger length using the Ngram-trigger. We set the pollution rate at 1‰ and use a range of ngrams with increasing lengths starting from one to four ("m", "ms", "mix" and "moni"). As mentioned earlier, we only evaluate the ASR of the backdoor attack model, and Figure 8 shows the results. As the figure shows, our backdoor attack is able to achieve an attack success rate of 100% when the length of ngram is great than 1. When the length of the trigger is increased from 1 to 2, the attack success rate of the model increases from 7% to nearly 100%. However, the trigger length continues to increase from 2, and there is no significant change. Therefore, in the BadNgram attack, a trigger with a length of 2 is a good choice.

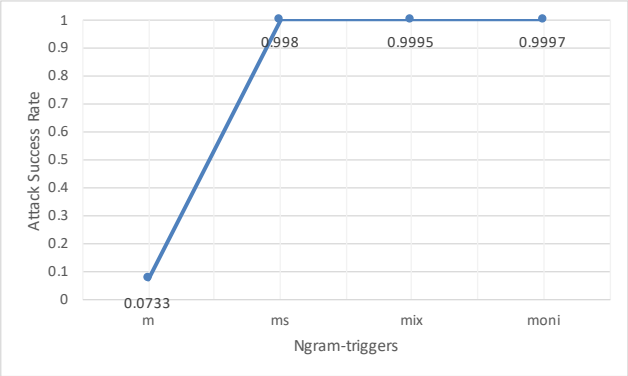


Figure 8. The trigger length evaluation of the Ngram-triggers.

7. Possible Defense

Given the severe harm of backdoor attacks, many defense measures have been proposed by researchers [44–49], but these methods can not be directly applied to the DGA field. For BadDGA, we propose two possible defenses.

Alert to abnormal data. The dataset polluted by the backdoor attack will differ from the clean dataset. After BadTLD injection, particular uncommon TLDs will appear in large numbers in the dataset, such as the domain name with ".bt" as the TLD. After BadNgram and BadWord injection, there will be many domain names containing the same ngram in the training dataset. In addition, we need to pay more attention to IDN-type domain names.

Contextual information is important. It is unreliable to rely only on the domain name characters' information, and context-related information is necessary. Context-related information, such as WHOIS information, domain name registration information, traffic information, etc., is difficult to be tampered with, so it has strong robustness.

8. Conclusions

In this paper, we propose, BadDGA, the first backdoor attack against an LSTM-based DGA detector in the black-box setting. Different from the previous backdoor attacks in the NLP field, four types of triggers are adopted to pollute the DGA training dataset. To evaluate the performance of our backdoor attack against the LSTM-based DGA detector, we use two metrics, including ASR and BA. Extensive experimental results show the effectiveness and stealthiness of BadDGA. In the future, we will design more excellent triggers and study backdoor attacks in the white-box mode. We hope our research can bring the attention of the security community to defend against such backdoor attacks on an LSTM-based DGA detector.

Author Contributions: Y.Z.: Methodology, Formal analysis, Software, and Original draft. L.Y.: Formal analysis, Software, Writing, and Editing. J.Y.: Review and Editing. L.H.: Review. Z.L.: Review and Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62276017, U1636211, 61672081), the Fund of the Key Laboratory of Power Grid Automation of China Southern Power Grid Co., Ltd. (Grant No. GDDKY2021KF03), and the Fund of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2021ZX-18).

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Obaidat, I.; Sridhar, M.; Pham, K.M.; Phung, P.H. Jadeite: A novel image-behavior-based approach for Java malware detection using deep learning. *Comput. Secur.* **2022**, *113*, 102547. [CrossRef]
2. Han, D.; Wang, Z.; Chen, W.; Zhong, Y.; Wang, S.; Zhang, H.; Yang, J.; Shi, X.; Yin, X. DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications. In Proceedings of the CCS 2021, Virtual, 15–19 November 2021; pp. 3197–3217.
3. Oh, S.E.; Yang, T.; Mathews, N.; Holland, J.K.; Rahman, M.S.; Hopper, N.; Wright, M. DeepCoFFEA: Improved Flow Correlation Attacks on Tor via Metric Learning and Amplification. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022; pp. 1915–1932.
4. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting domain generation algorithms with long short-term memory networks. *arXiv* **2016**, arXiv:1611.00791.
5. Fu, C.; Li, Q.; Shen, M.; Xu, K. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. In Proceedings of the ACM CCS 2021, Virtual, 15–19 November 2021; pp. 3431–3446.
6. Jianbing, L.; Shuhui, C.; Ziling, W.; Shuang, Z.; Wei, Z. HAGDetector: Heterogeneous DGA domain name detection model. *Comput. Secur.* **2022**, *120*, 102803.
7. Kundu, P.P.; Truong-Huu, T.; Chen, L.; Zhou, L.; Teo, S.G. Detection and Classification of Botnet Traffic using Deep Learning with Model Explanation. *IEEE Trans. Dependable Secur. Comput.* **2022**, *early access*.
8. Morbidoni, C.; Spalazzi, L.; Teti, A.; Cucchiarelli, A. Leveraging n-gram neural embeddings to improve deep learning DGA detection. In Proceedings of the SAC 2022, Virtual, 25–29 April 2022; pp. 995–1004.
9. Nguyen, T.N.; Ngo, Q.D.; Nguyen, H.T.; Giang, N.L. An Advanced Computing Approach for IoT-Botnet Detection in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2022**, *18*, 8298–8306. [CrossRef]
10. Tran, D.; Mac, H.; Tong, V.; Tran, H.A.; Nguyen, L.G. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* **2018**, *275*, 2401–2413. [CrossRef]
11. Xia, P.; Niu, H.; Li, Z.; Li, B. Enhancing Backdoor Attacks with Multi-Level MMD Regularization. *IEEE Trans. Dependable Secur. Comput.* **2022**, *early access*.
12. Xue, M.; He, C.; Wu, Y.; Sun, S.; Zhang, Y.; Wang, J.; Liu, W. PTB: Robust physical backdoor attacks against deep neural networks in real world. *Comput. Secur.* **2022**, *118*, 102726. [CrossRef]
13. Zhao, Z.; Chen, X.; Xuan, Y.; Dong, Y.; Wang, D.; Liang, K. DEFEAT: Deep Hidden Feature Backdoor Attacks by Imperceptible Perturbation and Latent Representation Constraints. In Proceedings of the CVPR 2022, New Orleans, LA, USA, 18–24 June 2022; pp. 15213–15222.
14. Wang, Z.; Zhai, J.; Ma, S. BppAttack: Stealthy and Efficient Trojan Attacks against Deep Neural Networks via Image Quantization and Contrastive Adversarial Learning. In Proceedings of the CVPR 2022, New Orleans, LA, USA, 18–24 June 2022; pp. 15074–15084.
15. Feng, L.; Li, S.; Qian, Z.; Zhang, X. Stealthy Backdoor Attack with Adversarial Training. In Proceedings of the ICASSP 2022, Singapore, 22–27 May 2022; pp. 2969–2973.
16. Phan, H.; Xie, Y.; Liu, J.; Chen, Y.; Yuan, B. Invisible and Efficient Backdoor Attacks for Compressed Deep Neural Networks. In Proceedings of the ICASSP 2022, Singapore, 22–27 May 2022; pp. 96–100.
17. Doan, K.; Lao, Y.; Zhao, W.; Li, P. LIRA: Learnable, Imperceptible and Robust Backdoor Attacks. In Proceedings of the ICCV 2021, Montreal, QC, Canada, 10–17 October 2021; pp. 11946–11956.
18. Chen, X.; Salem, A.; Chen, D.; Backes, M.; Ma, S.; Shen, Q.; Wu, Z.; Zhang, Y. BadNL: Backdoor Attacks against NLP Models with Semantic-preserving Improvements. In Proceedings of the ACSAC 2021, Online, 6–10 December 2021; pp. 554–569.
19. Bagdasaryan, E.; Shmatikov, V. Spinning Language Models: Risks of Propaganda-as-a-Service and Countermeasures. In Proceedings of the SP 2022, San Francisco, CA, USA, 22–26 May 2022; p. 1532.
20. Liu, Y.; Shen, G.; Tao, G.; An, S.; Ma, S.; Zhang, X. PICCOLO: Exposing Complex Backdoors in NLP Transformer Models. In Proceedings of the SP 2022, San Francisco, CA, USA, 22–26 May 2022; p. 1561.
21. Qi, F.; Li, M.; Chen, Y.; Zhang, Z.; Liu, Z.; Wang, Y.; Sun, M. Hidden Killer: Invisible Textual Backdoor Attacks with Syntactic Trigger. In Proceedings of the ACL/IJCNLP 2021, Bangkok, Thailand, 1–6 August 2021; Volume 1, pp. 443–453.
22. Yang, W.; Lin, Y.; Li, P.; Zhou, J.; Sun, X. Rethinking Stealthiness of Backdoor Attack against NLP Models. In Proceedings of the ACL/IJCNLP 2021, Bangkok, Thailand, 1–6 August 2021; Volume 1; pp. 5543–5557.
23. Shao, K.; Zhang, Y.; Yang, J.; Li, X.; Liu, H. The triggers that open the NLP model backdoors are hidden in the adversarial samples. *Comput. Secur.* **2022**, *118*, 102730. [CrossRef]
24. Gan, L.; Li, J.; Zhang, T.; Li, X.; Meng, Y.; Wu, F.; Yang, Y.; Guo, S.; Fan, C. Triggerless Backdoor Attack for NLP Tasks with Clean Labels. In Proceedings of the NAACL 2022, Seattle, WA, USA, 10–15 July 2022; pp. 2942–2952.
25. Yang, J.; Yin, Y.; Ma, S.; Huang, H.; Zhang, D.; Li, Z.; Wei, F. Multilingual Agreement for Multilingual Neural Machine Translation. In Proceedings of the ACL 2021, Bangkok, Thailand, 1–6 August 2021; pp. 233–239.
26. Yang, J.; Ma, S.; Huang, H.; Zhang, D.; Dong, L.; Huang, S.; Muzio, A.; Singhal, S.; Hassan, H.; Song, X.; et al. Multilingual Machine Translation Systems from Microsoft for WMT21 Shared Task. In Proceedings of the WMT 2021, Online, 10–11 November 2021; pp. 446–455.

27. Wang, H.; Xu, T.; Yang, J.; Wu, L.; Yang, L. Sessionvideo: A Novel Approach for Encrypted Traffic Classification via 3D-CNN Model. In Proceedings of the APNOMS 2022, Takamatsu, Japan, 28–30 September 2022; pp. 1–6.
28. Yang, J.; Yin, Y.; Ma, S.; Zhang, D.; Li, Z.; Wei, F. High-resource Language-specific Training for Multilingual Neural Machine Translation. In Proceedings of the IJCAI 2022. ijcai.org, Vienna, Austria, 23–29 July 2022; pp. 4461–4467.
29. Yang, J.; Ma, S.; Zhang, D.; Li, Z.; Zhou, M. Improving Neural Machine Translation with Soft Template Prediction. In Proceedings of the ACL 2020, Online, 5–7 July 2020; pp. 5979–5989.
30. Mockapetris, P.V. *Domain Names: Implementation Specification*; Technical Report; 1983. Available online: https://dl.acm.org/doi/book/10.17487/RFC1035 (accessed on 1 January 2023).
31. Mockapetris, P.V. *Domain Names-Concepts and Facilities*; Technical Report; 1987. Available online: https://datatracker.ietf.org/doc/rfc1034/ (accessed on 1 January 2023).
32. Geffner, J. End-to-end analysis of a domain generating algorithm malware family. In Proceedings of the Black Hat 2013, San Francisco, CA, USA, 24–28 February 2013.
33. Plohmann, D.; Yakdan, K.; Klatt, M.; Bader, J.; Gerhards-Padilla, E. A comprehensive measurement study of domain generating malware. In Proceedings of the USENIX Security 2016, AUSTIN, TX, USA, 10–12 August 2016; pp. 263–278.
34. Gu, T.; Dolan-Gavitt, B.; Garg, S. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv* **2017**, arXiv:1708.06733.
35. Chen, X.; Liu, C.; Li, B.; Lu, K.; Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv* **2017**, arXiv:1712.05526.
36. Schüppen, S.; Teubert, D.; Herrmann, P.; Meyer, U. FANCI: Feature-based Automated NXDomain Classification and Intelligence. In Proceedings of the USENIX Security 2018, Baltimore, MD, USA, 5–17 August 2018; pp. 1165–1181.
37. Pereira, M.; Coleman, S.; Yu, B.; DeCock, M.; Nascimento, A. Dictionary extraction and detection of algorithmically generated domain names in passive DNS traffic. In Proceedings of the RAID 2018, Heraklion, Greece, 10–12 September 2018; pp. 295–314.
38. Yu, B.; Pan, J.; Hu, J.; Nascimento, A.; De Cock, M. Character level based detection of DGA domain names. In Proceedings of the IJCNN 2018, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
39. Huang, J.; Wang, P.; Zang, T.; Qiang, Q.; Wang, Y.; Yu, M. Detecting domain generation algorithms with convolutional neural language models. In Proceedings of the TrustCom/BigDataSE 2018, New York, NY, USA, 1–3 August 2018; pp. 1360–1367.
40. Dai, J.; Chen, C.; Li, Y. A backdoor attack against lstm-based text classification systems. *IEEE Access* **2019**, *7*, 138872–138878. [CrossRef]
41. Kurita, K.; Michel, P.; Neubig, G. Weight poisoning attacks on pre-trained models. *arXiv* **2020**, arXiv:2004.06660.
42. Chan, A.; Tay, Y.; Ong, Y.; Zhang, A. Poison Attacks against Text Datasets with Conditional Adversarially Regularized Autoencoder. In Proceedings of the EMNLP 2020, Virtual Conference, 16–20 November 2020; pp. 4175–4189.
43. Qinge, X.; Shujun, T.; Xiaofeng, Z.; Qingran, L.; Baojun, L.; Haixin, D.; Frank, L. Building an Open, Robust, and Stable Voting-Based Domain Top List. In Proceedings of the USENIX Security 2022, Boston, MA, USA, 10–12 August 2022; pp. 625–642.
44. Lyu, W.; Zheng, S.; Ma, T.; Chen, C. A Study of the Attention Abnormality in Trojaned BERTs. In Proceedings of the NAACL 2022, Seattle, WA, USA, 2–4 March 2022; pp. 4727–4741.
45. Shao, K.; Yang, J.; Ai, Y.; Liu, H.; Zhang, Y. BDDR: An Effective Defense Against Textual Backdoor Attacks. *Comput. Secur.* **2021**, *110*, 102433. [CrossRef]
46. Yang, W.; Lin, Y.; Li, P.; Zhou, J.; Sun, X. RAP: Robustness-Aware Perturbations for Defending against Backdoor Attacks on NLP Models. In Proceedings of the EMNLP 2021, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 8365–8381.
47. Fan, M.; Si, Z.; Xie, X.; Liu, Y.; Liu, T. Text Backdoor Detection Using an Interpretable RNN Abstract Model. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4117–4132. [CrossRef]
48. Qiu, H.; Zeng, Y.; Guo, S.; Zhang, T.; Qiu, M.; Thuraisingham, B. DeepSweep: An Evaluation Framework for Mitigating DNN Backdoor Attacks using Data Augmentation. In Proceedings of the ASIA CCS 2021, Hong Kong, China, 7–11 June 2021; pp. 363–377.
49. Doan, B.G.; Abbasnejad, E.; Ranasinghe, D.C. Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. In Proceedings of the ACSAC 2020, Online, 7–11 December 2020; pp. 897–912.