# Quicksolver: A lightweight malicious domains detection system based on adaptive autoencoder

Wei Wan*†‡ , Ke Wang†‡ , Jinxia Wei†‡ , Liangyi Gong‡ , Pan Huang‡ , Hao Fu†‡, and Yuhao Fu‡
*Corresponding Author: Wei Wan Email: anquanip@cnic.cn
†Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
‡University of Chinese Academy of Sciences, Beijing, China

2023 IEEE 48th Conference on Local Computer Networks (LCN) | 979-8-3503-0073-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/LCN58197.2023.10223356

***Abstract*—The Domain Name System (DNS) plays a critical role in the Internet, making it a popular target for cyber attackers. Malicious actors use DNS to locate their command and control servers, and spam often contains URLs linked to domains that host malicious servers. Detecting such malicious domain activities is essential. While many prior works have shown promising results in detecting malicious domains, the time and storage required during detection are relatively high.**

**In this paper, we propose a lightweight and effective malicious domain detection system called Quicksolver, ideal for large-scale networks. Our system uses only domain features, eliminating the need for additional costs associated with DNS traffic and registration information. Additionally, we use an improved autoencoder as our classifier, combining it with neural networks to avoid the need for setting and adjusting thresholds manually.**

**We evaluated Quicksolver using malicious data collected from a certain ISP over three months. The results show that Quicksolver has better detection ability and lower detection time compared to other state-of-the-art methods. Furthermore, it can automatically identify unknown malicious domains that are misused in seven types of cyber attacks.**

***Index Terms*—Malicious domain detection, Autoencoder, Network security**

## I. Introduction

The Domain Name System (DNS) is a critical Internet service that maps IP addresses and domain names to each other. While IP addresses are unique identifiers in network communications, their numerical format can be difficult for users to remember, which is why domain names are widely used on the Internet. However, the DNS has also become a major avenue for various cyber attacks.

Numerous researchers have conducted studies on methods to detect malicious domain names based on features calculated apart from domains. Some studies have implemented malicious domain name detection based on DNS traffic analysis [1]–[4], where they monitor a period of DNS traffic and use the correlation between normal domain names to detect malicious domains. Other researchers rely on a large amount of domain name registration information to detect malicious domains during the registration stage [5]–[7]. While these prior works gather abundant information from domains and show promising detection capabilities, they require additional time and memory costs before detection. In particular, storing DNS request traffic for a period of time in a large-scale backbone network will occupy a lot of memory, and querying Whois also requires time in a large-scale network.

While previous research has shown good detection results by extracting domain-related features from multiple perspectives, the application of these methods in large-scale networks is limited due to high overheads of time and memory. In particular, large-scale backbone networks have high demands on the time and storage overhead of domain name detection systems. If the detection system incurs high time costs and does not use storage, many DNS requests may be lost, potentially leading to malicious domain name requests being discarded. Conversely, if the system stores undetected requests, more and more unprocessed DNS requests will take up a lot of memory over time. Considering the application of malicious domain detection systems in large-scale scenarios, we want to design a malicious detection system from two considerations, high detection capability, and low overhead both in time and storage.

In order to build an efficient and effective malicious domain detection system for large-scale networks, this study extracted domain-related features that distinguish between normal and malicious domains. To minimize overhead, priority was given to features that do not require additional query time and storage. The study was inspired by features used in related work [1], [5], [8], and further analysis of the malicious domain names in the data set revealed that normal domains are simple, readable, and understandable, while malicious domains are randomly generated by DGA to avoid being blacklisted. Based on these observations, the study calculated features from domain structure and semantic perspective. The K-S [9] test showed that the p-values of all the aforementioned features were less than 0.05, indicating significant differences between normal and malicious domains in terms of these features.

After feature extraction, a detection model is needed to automatically and effectively detect malicious domains. Several classical machine learning classification models, including One Class SVM [8], SVM [10], Bayes [11], K-means [12], ANN [13], and autoencoder [14], are compared based on the calculated features. Autoencoder is found to have better anomaly detection ability than other models and is chosen as the basic detection model. However, the autoencoder requires manual setting of thresholds, which is time-consuming and not scalable. To overcome this limitation, ANN and autoencoder are combined to detect malicious domains without setting a threshold. The ANN updates itself based on recent data to adapt to the current network scenarios.

To evaluate the detection ability of the Quicksolver system in real-life, it is applied to a large, real-world dataset collected over a period of three months in a certain ISP. The dataset includes 140,000 malicious domains and covers seven types of attacks. The results show that Quicksolver has a better detection ability (F1-score:92.05%) than n-CBDC (F1-score:73.65%) [15], a method based on the distribution of domains' character distribution, and Premadoma (F1-score: 35.34%) [5], a typical method based on domains' registration information. Furthermore, Quicksolver takes less detection time (7424 domains/s) than n-CBDC (746 domains/s) and Premadoma (11 domains/s).

In summary, our paper makes following contributions:

- Compared to methods that extract features from sources other than domains (such as DNS traffic over a period of time and registration information), our approach extracts features directly from domains without additional costs (such as storage for storing DNS traffic or detection of delay time caused by WHOIS queries).
- An improved autoencoder is proposed as a detection model, which eliminates the need to manually set and adjust the threshold by combining it with an ANN. This results in a reduction of labor costs associated with threshold setting and an improvement in detection capability.
- We propose a lightweight malicious detection system called Quicksolver, which has been evaluated using real-world data collected from a certain ISP over a period of three months. Our results indicate that Quicksolver outperforms other methods in terms of higher precision and lower time costs. Additionally, our approach is more generic and not limited to specific types of attacks, such as bots.

## II. System

The Quicksolver aims to develop a malicious domain detection system that is low-cost, high-accuracy, and effective in unbalanced data and dynamic environments. To implement malicious domain detection, two steps must be followed. One is features extraction, another is detection model selection according to requirement.

*1) Feature extraction:* Unlike previous studies, some additional information (e.g. .large amount of registration information, DNS traffic) [1]–[7] is required to extract features. This paper extract features from domain names with low computational overhead and no additional information required, making it more suitable for malicious domain detection tasks in large-scale backbone networks. The features used in the detection model include frequency of TLD, the length of domain, the entropy of domain, the frequency of number, the level of subdomain, frequency of vowel, randomness, mean and variance of 1-gram to 3-gram. After extracting the features from the domains, the K-S [9] test is carried and the p-values of all the aforementioned features were less than 0.05, indicating significant differences between normal and malicious domains in terms of these features.

TABLE I
Classification results from different detection model

| Model | ACC | Pre | Recall | F1 |
|---|---|---|---|---|
| One Class SVM [8] | 0.7903 | 0.9085 | 0.7135 | 0.7993 |
| SVM [10] | 0.9110 | 0.8558 | 0.7164 | 0.7799 |
| Bayes [11] | 0.6746 | 0.7807 | 0.6176 | 0.6896 |
| K-means [12] | 0.5087 | 0.9945 | 0.1615 | 0.2779 |
| ANN [13] | 0.9307 | 0.9261 | 0.7446 | 0.8255 |
| Autoencoder [14] | 0.7368 | 0.7013 | 0.8771 | 0.7795 |

*2) Detection model selection:* Malicious domain detection is a critical task in cybersecurity, and it is essentially an anomaly detection task. In contrast to general classification tasks, anomaly detection tasks often face data imbalance issues, where the number of normal samples far exceeds that of anomaly samples. This imbalance can cause bias in supervised machine learning models and result in poor anomaly detection performance. Therefore, unsupervised models are commonly used in anomaly detection tasks, with the autoencoder being a representative model of unsupervised deep learning widely used in various anomaly detection applications [16].

In this paper, we evaluate the detection ability of commonly used classification models for malicious domain detection using domain names collected from a certain ISP. The results in Table I show that the autoencoder outperforms other classification models in recall, indicating its superior ability to detect malicious samples. Thus, we choose the autoencoder as the basic detection model in this paper.

The autoencoder is an unsupervised deep learning model comprising an encoder, a decoder, and a hidden layer. It compresses high-dimensional data into the hidden layer to learn representational features, and then reconstructs the data through the decoder. The objective is to minimize the mean square error between the output and input. In anomaly detection, the trained autoencoder calculates the reconstruction error for test samples and compares it with a threshold value. However, manually setting the threshold can impact detection capability. Experiments reveal that increasing the threshold raises precision but lowers recall, increasing false alarms and reducing misses. Conversely, decreasing the threshold reduces false alarms but increases misses. Hence, we propose an improved autoencoder-based detection model that eliminates manual threshold setting, described in detail in the following section.

To avoid the manual setting of threshold, we propose a method that combines autoencoder with neural network. In the training phase, the autoencoder is trained with normal samples, and the reconstruction error of the test samples is calculated using the trained autoencoder. Subsequently, the reconstruction error and labels are fed into the neural network, where the weight of the reconstruction error in each dimension is adjusted during the training process. This adjustment results in the final reconstruction error of abnormal samples converging to 1 and the final reconstruction error of normal samples converging to 0, thus improving the differentiation

between normal and abnormal samples. During the detection phase, the neural network is used to automatically classify the test samples, thereby avoiding manual threshold setting and improving the detection model's capability.

*3) System Architecture:* As shown in Fig. 1, our detection system can be divided into two phases, an offline training and an online detection phase.

The main objective of the training phase is to obtain a trained autoencoder. The data used consists of normal domain name data. Firstly, the domain names undergo cleaning to filter out those containing IP addresses and pure numeric domain names. Cleaning the dataset helps the subsequent model learn the distribution of normal domain names more effectively. After cleaning, features are extracted for each domain and used as input for the detection model during the training phase. The goal is to obtain a well-trained autoencoder model that can learn the distribution of normal samples accurately. In the detection phase, the trained model calculates the reconstruction error of test samples and measures the difference between normal and malicious samples based on the reconstruction error under different features.

The testing phase consists of two models, a trained autoencoder and a classifier based on a neural network with adaptive capabilities. For the domains to be detected, the features are first extracted, the feature vector is used as input to the autoencoder to calculate the reconstruction error in each dimension, and subsequently, the classifier module gives the detection result.

## III. Evaluation

### A. Evaluation detection ability on real-world data set

*1) Dataset:* The data set used to evaluate the performance of the detection model in this paper consists of normal and abnormal domains. The normal domains are provided by intelligence provider. While malicious domains are from a certain 100Gbps backbone ISP with more than 1 million network users. We collected about 140,000 malicious domains from January 5 to April 29 in 2022 and they cover eight attacks including phishing, spam, malware, worms, Trojan, botnet, mining, and Virus. And spam, phishing, worm, malware, trojan, mining, botnet, and virus accounted for 34.6%, 24.6%, 20.51%, 12.73%, 4.57%, 1.24% and 0.04% respectively.

*2) Experimental environment and evaluation metric:* This experiment runs on an Intel(R) Core(TM) i7-7700 CPU@3.6 GHz with 16 GB RAM on a Windows 10 system and a development environment of PyCharm with Anaconda3 (Python3. 7). In this paper, we use the classical evaluation metrics in machine learning, including accuracy, precision, recall, and F1-Score.

*3) Comparisons with other detection systems:* In this paper, we conducted an extensive review of studies on malicious domains and identified n-CBDC [15] as a representative method based on character distribution and Premadoma [5] as a typical method based on feature extraction from domains and domain registration. Comparison experiments were carried out and the results are presented in Table II. It was observed that

TABLE II
THE RESULTS OF DIFFERENT MALICIOUS DOMAINS DETECTION METHODS

| Method | Accruacy | Precision | Recall | F1-socre |
|---|---|---|---|---|
| Quicksolver | 91.88% | 95.88% | 88.51% | 92.05% |
| n-CBDC [15] | 80.18% | 81.82% | 66.97% | 73.65% |
| Premadoma [5] | 87.45% | 89.94% | 83.44% | 86.57% |

Quicksolver outperformed both comparison methods in terms of accuracy, precision, recall, and f1-score. Following the original paper's idea [15], second-level domains were extracted and encoded using n-grams. However, it was found that 9.3% of malicious domains' second-level domains were also present in normal domains, making it impossible to classify domain names as normal or abnormal by solely detecting second-level domain names. Quicksolver, on the other hand, targets the entire domain and includes not only character distribution but also domain structure features. In the paper [5], three types of features were utilized, namely, domain features, registration features, and reputation scores. However, the reputation scores were computed using a private database and are not reproducible on realistic datasets. Therefore, in this paper, only domain and registration features were used for the experiment. Nevertheless, due to the presence of numerous missing values, the feature vector cannot differentiate between normal and abnormal effectively. Conversely, Quicksolver features are derived solely from the domain, with no missing values, and provide better differentiation between normal and abnormal. Furthermore, while the method in the original paper employed a traditional random forest model, the detection model proposed in this paper is based on deep learning. By iteratively adjusting the model parameters, deep learning can learn the deeper representation of the dataset more effectively than traditional machine learning models, resulting in a higher detection capability.

### B. Evaluate detection efficiency on real-life network

Quicksolver is implemented in a certain ISP for about one day. The DNS requests of it are about 100 million in one hour, up to 149 million, with a minimum of 57 million. In realistic detection scenarios, when the DNS requests acceptance rate exceeds the processing speed of the detection model, the redundant requests will be dropped. In order to present different detection methods process speed, we use the processed rate which is the quotient of the number of accepted requests divided by the total DNS requests. The processed rate of different detection methods is shown in Fig. 2, the dot in each line present the processed rate in certain hour in certain methods. We can see from the Fig. 2, the Premadoma processed rate comes close to 0 because the time costs in getting registration information are too much. And the n-CDBC is come close to 0.15, the Quicksolver is around about 0.25. It is obvious that time costs in Quicksolver is slower than other method. With so many DNS requests in backbone network(about 100 million), we used multi-thread to process so many data and calculated processed rate. As is shown in the Fig. 2, the processed rate increases as the number of threads

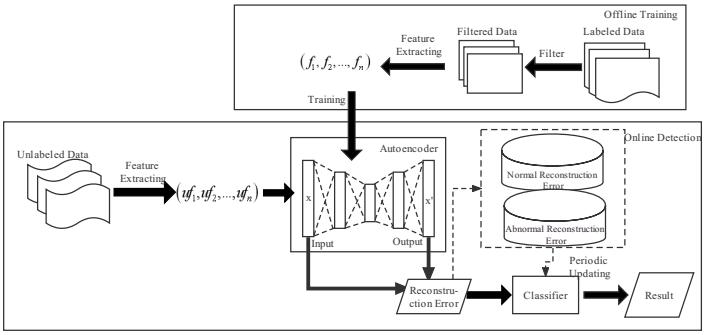


Fig. 1. The architecture of Quicksolver

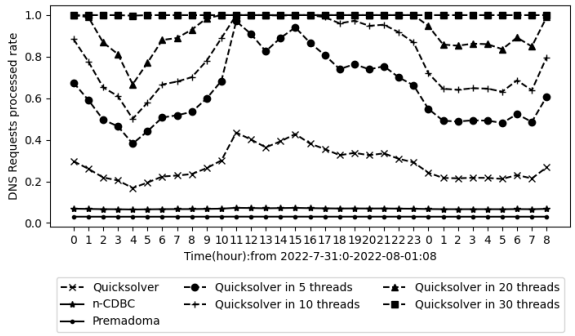


Fig. 2. The DNS requests processed rate of different methods

increases. When thirty threads are used, the processed rate comes nearly close to 1. We can say our detection model can be applied to large scale network.

## IV. Conclusion

In this work, we presented Quicksolver, a lightweight system for the detection of malicious based on improved autoencoder without labor cost in setting the threshold and effective and efficient features relying exclusively on domain names. In our extensive evaluation, we verified Quicksolver's highly accurate and highly efficient detection capabilities in the real-life data set. Thus it can contribute to network security in a variety of environments including large-scale networks.

In the future work, not only will we know if the domain name is malicious, but also what kind of attack the malicious domain is associated with. Also we want to know which features lead to domain name being judged as such an attack to increase the interpretability of the model.

## References


[1] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of dga-based malware," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 491–506.

[2] X. Sun, M. Tong, J. Yang, L. Xinran, and L. Heng, "Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, 2019, pp. 399–412.

[3] S. Zhang, Z. Zhou, D. Li, Y. Zhong, Q. Liu, W. Yang, and S. Li, "Attributed heterogeneous graph neural network for malicious domain detection," in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2021, pp. 397–403.

[4] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/Acm Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[5] J. Spooren, T. Vissers, P. Janssen, W. Joosen, and L. Desmet, "Premadoma: An operational solution for dns registries to prevent malicious domain registrations," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 557–567.

[6] T. Vissers, J. Spooren, P. Agten, D. Jumpertz, P. Janssen, M. V. Wesemael, F. Piessens, W. Joosen, and L. Desmet, "Exploring the ecosystem of malicious domain registrations in the. eu tld," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2017, pp. 472–493.

[7] S. Hao, A. Kantchelian, B. Miller, V. Paxson, and N. Feamster, "Predator: proactive recognition and elimination of domain abuse at time-of-registration," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1568–1579.

[8] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[9] G. Fasano and A. Franceschini, "A multidimensional version of the kolmogorov–smirnov test," *Monthly Notices of the Royal Astronomical Society*, vol. 225, no. 1, pp. 155–170, 1987.

[10] M. Pal and G. M. Foody, "Feature selection for classification of hyperspectral data by svm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, pp. 2297–2307, 2010.

[11] H.-C. Kim and Z. Ghahramani, "Bayesian classifier combination," in *Artificial Intelligence and Statistics*. PMLR, 2012, pp. 619–627.

[12] J. Xu and K. Lange, "Power k-means clustering," in *International conference on machine learning*. PMLR, 2019, pp. 6921–6931.

[13] S. S. Haykin *et al.*, "Neural networks and learning machines/simon haykin." 2009.

[14] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[15] C. Xu, J. Shen, and X. Du, "Detection method of domain names generated by dgas based on semantic representation and deep neural network," *Computers & Security*, vol. 85, pp. 77–88, 2019.

[16] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.