IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 July 2022, accepted 8 August 2022, date of publication 16 August 2022, date of current version 22 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3198687

RESEARCH ARTICLE

Detection Method for Randomly Generated User IDs: Lift the Curse of Dimensionality

INWOO RO[1,3], BOOJOONG KANG[2], CHOONGHYUN SEO[1], AND EUL GYU IM[3]

[1]NAVER WEBTOON Ltd., Seongnam, Gyeonggi 13561, South Korea
[2]School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.
[3]Department of Computer Science, Hanyang University, Seoul 04763, South Korea

Corresponding author: Eul Gyu Im (imeg@hanyang.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant through the Korea Government [Ministry of Science and ICT (MSIT)], Decentralized High Performance Consensus for Large-Scale Blockchains, under Grant 2021-0-00590.

ABSTRACT Internet services are essential to our daily life in these days, and user accounts are usually required for downloading or browsing for multimedia contents from service providers such as Yahoo, Google, YouTube and so on. Attackers who perform malicious actions against these services use fake user accounts to hide their identity, or use them to continue malicious actions even after being caught by the service's detection system. Using a random string generation algorithm for user identification (ID) string is one of the common method to create and obtain a large number of fake user accounts. To detect IDs and to defend against such attacks, some researchers have proposed the models that detect randomly generated IDs. Among these detection models, the *n*-gram-based using term frequency-inverse document frequency model is regarded as a state-of-the-art model to detect randomly generated IDs, but *n*-gram-based approaches have the problem of the curse of dimensionality because the sparsity of feature vector increases exponentially with the increase of size *n*. As a result, the improvement of the detection accuracy is limited since size *n* cannot be increased. This paper proposes two methods to detect randomly generated IDs more accurately. The first is to avoid the curse of dimensionality with the compression of feature dimension size. The second is a technique to reduce false positives by using pattern matching and Bhattacharyya distance. We tested our method with about 3 million normal user IDs collected from the real portal service, 1 million IDs generated by a random string generation algorithm, and 8,541 IDs found after being used for malicious behavior in real portal services. The experimental results showed that the proposed method can improve detection accuracy as well as inference performance.

INDEX TERMS Authentication, computer crime, identity management systems, web sites.

I. INTRODUCTION

Fake user accounts are used for multiple bot operations, Social Network Service (SNS) manipulation, and contents piracy. Examples of malicious behavior using these fake accounts include a case where a botnet was constructed and used to manipulate Twitter public opinion [1]. In case of contents piracy, there are piracy sites that generate revenue by uploading copyright contents without permission of the owner [2]. If a malicious user downloads content with a single user ID, a large amount of access logs are left in proportion to the number of accessed content. A service provider can detect the malicious ID simply by examining the access logs and can find the ID that is not in normal range [3]. To circumvent the ID detection, attackers need to generate and register a larger number of IDs to access and download multimedia contents. There are attackers generate a number of IDs using a random string generation algorithm to register IDs as many as possible, so suspicious IDs can be detected by examining whether an ID contains a random string.

Various methods have been proposed to detect suspicious IDs and the previous approaches can be divided into two

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

categories. The first category is featureless methods which analyzes strings directly through a deep learning model without a feature extraction process. A common problem with these techniques is that it is difficult to explain the causes of the results of deep learning models. The other is feature-based methods which extract features to detect suspicious IDs. One of the latest methods is the *n*-gram-based randomly generated ID detection method using term frequency–inverse document frequency (TF-IDF), proposed by Beskow and Carley [22]. However, their method suffers from the curse of dimensionality because the feature dimension increases exponentially with increasing *n* in *n*-gram-based approaches.

This paper proposes a new method that compresses TF-IDF term vectors to avoid the problem of the curse of dimensionality. We also found that benign user IDs with rare ID strings are being detected falsely as randomly generated IDs, even after solving the curse of dimensionality. To reduce such false positives, this paper proposes another feature that compares the alphabetical distributions of user IDs using the Bhattacharyya distance [4].

The contributions of this paper are as follows:

- We solve the sparsity problem by compressing the information for the TF-IDF term vector. Through this, we propose a technique that does not have the disadvantages of featureless models while overcoming the limitations of existing feature-based models.
- We propose a new feature by using the difference in the alphabetical distribution of randomly generated user IDs and normal user IDs.
- Experimental results on a real dataset compromising more than 3 million normal IDs and 8,541 randomly generated user IDs that were discovered after actually engaging in malicious behavior confirmed the proposed algorithm achieved 99.15% accuracy with 1.5 % false positive rate.

The rest of this paper is organized as follows: Section II presents the literature review and Section III discusses the problems of the state of the art. Section IV proposes our method that tackles the problems. Section V describes our evaluation and analyzes the results. Section VI concludes the paper.

II. RELATED WORK

Cyber security defense systems often include a mechanism for detecting randomly generated strings which are used in cyber attacks. Botnets equip with domain generation algorithms (DGAs) that produce a set of domain names to stay hidden behind a different domain name at a different time. Malicious users use random strings for user IDs to create massive fake accounts.

Various detection mechanisms allow cyber security defenders to locate where cyber attacks happen and trigger further action(s). In this section, we categorize machine learning based methods into two approaches: featureless and feature-based approaches and describes strengths and weaknesses of each approach.

Our proposed technique belongs to the feature based approach category. However, this technique overcomes the limitations of the feature based approach category by solving the curse of dimensionality problem.

A. FEATURELESS APPROACHES

Featureless approaches have no feature extraction step and directly analyze the string of domain names to identify if it is malicious. Woodbridge *et al.* [5] proposed the first deep learning approach and showed that character level long short-term memory networks (LSTMs) could effectively identify DGAs. Yu *et al.* [6] proposed the first convolutional neural networks (CNNs) based approach and a subsequent study [7] showed that a parallel CNN architecture improved the detection accuracy. Berman *et al.* [8] proposed a new architecture with a capsule neural network (CapsNet) and Qiao *et al.* [9] investigated attention mechanisms in LSTM. Yang *et al.* [11] proposed a heterogeneous deep neural network framework comprising an improved parallel CNN (IPCNN) and self-attention based bidirectional LSTMs. Ma *et al.* [12] proposed a new detection method based on Doc2vec and hybrid network (LSTM-RNN).

This deep neural network-based featureless methods has a problem in terms of interpretability. As Feng *et al.* [10] pointed out, deep learning methods have an unexplainable nature that works as a black box model. Identifying Fake User Accounts may lead to blocking and legal measures. Therefore, it can be a fatal disadvantage the fact that the grounds of identification cannot be explained can be a fatal disadvantage.

B. FEATURE BASED APPROACHES

Feature based approaches extract features from the string of domain names to model benign and malicious domain names with machine learning algorithms. *N*-gram extracts consecutive subsequences from a given string and has been widely adopted in many string analysis methods. Yadav *et al.* [13] extracted uni and bi-grams from domain names and measured Kullback-Leibler (KL) divergence, Jaccard coefficient, and edit distance to characterize domain names that share the same IP address. Cucchiarelli *et al.* [14] extracted bi and 3-grams, and measured KL divergence and Jaccard coefficient. Antonakakis *et al.* [15] extracted up to 4-grams and compute the median, average and standard deviation of *n*-gram frequency distributions to compare domain names in unsuccessful domain name service requests. Schiavoni *et al.* [16] extracted up to 3-grams and counted their occurrence in a language dictionary to capture the pronounceability of domain names; defining domain names with a low count as probably DGA generated. Selvi *et al.* [17] converted every character of domain names into a symbol representing its type: a consonant 'c', a vowel 'v', a digit 'n', and any other 's'. *N*-grams, up to 4-grams, are extracted from the converted domain names and these special *n*-grams

are defined as masked *n*-grams. This technique reduces the number of extracted *n*-grams and also reduces the computational complexity significantly. Xu *et al.* [18] extracted uni- and bi-grams and adopted CNNs instead of similarity or distance measures such as Jaccard coefficient and edit distance. Alaeiyan *et al.* [19] extracted *n*-skip-grams, which are extracted from (*n*+2)-grams by discarding *n* center characters of the (*n*+2)-grams. For an example string 'string', 1-skip-grams are 'sr', 'ti', 'rn' and 'ig', and 2-skip-grams are 'si', 'tn' and 'rg'. *N*-gram has been used in many DGA detection methods but such *n*-gram-based methods have a scalability issue that the amount of data increases exponentially by the value of *n*. Although Selvi *et al.* [17] reduce the number of *n*-grams by a masking technique, information from original characters is discarded which might be still useful. Many DGA detection methods cannot be directly used for detecting fake user accounts because domain names and user IDs are not the same.

Freeman [20] proposed a Naive Bayesian model to detect spamming IDs in social networks. *N*-gram is used to extract features from letters of users' first and last names. The model was evaluated with different values of *n*, up to 5, and there was a trade-off between the accuracy and the computational costs. When a higher value of *n* was used, the model was more accurate for detecting spamming IDs but it took a longer time. Beskow and Carley [22] proposed a randomly generated user ID detection method based on its randomness of strings in user IDs. Many fake user accounts are randomly generated and such randomly generated IDs are likely to have rare combinations of characters. TF-IDF is employed to analyze how popular or rare each "term" is in user IDs. If a term appears in many other user IDs, TF-IDF will be a larger value and the term will be considered as popular and vice versa. *N*-gram is also employed to analyze variable lengths of terms in user IDs. By combining TF-IDF and *n*-gram, the popularity or rareness of terms with a variable length can be measured and used for characterizing a given string. The proposed method showed a good detection accuracy however the detection accuracy decreased when a higher value of *n* was used. *N*-gram with a higher value of *n* not only requires high computational costs but also leads to the curse of dimensionality [21] in machine learning-based solutions.

This paper investigates the effectiveness of *n*-gram with different values of *n* for randomly generated user ID detection and proposes a method to overcome the curse of dimensionality when a higher value of *n* is used. Through this approach, our methods overcomes the curse of dimensionality problem of the feature-based method without having an interpretability problem of the featureless method.

III. PROBLEM DEFINITION

This section defines the problems that this paper aims to tackle. Following subsections will discuss two problems: the curse of dimensionality and rare normal IDs.

(a) String	"abc-abc"
(b) 3-gram	["abc", "bc-", "c-a", "-ab"]
(c) 3-gram TF-IDF	[13.98, 12.75, 13.34, 12.87]
(d) 3-gram Term TF-IDF vector	[0⋯0,13.98,0⋯0,12.75, 0⋯0,13.34,0⋯0,12.87,0⋯0] ("abc", "bc-", "c-a", "-ab")

FIGURE 1. *N*-gram TF-IDF vector from an example string.

A. CURSE OF DIMENSIONALITY

The model proposed by Beskow and Carley [22] has a high false positive rate due to the curse of dimensionality when calculating the TF-IDF vector. Beskow's work extracts *n*-grams from a user ID string and applies TF-IDF to compute term vectors for the *n*-grams. Figure 1 shows an example user ID "*abc-abc*", where:

- 3-grams for the string = { '*abc*', '*bc-*', '*c-a*', '*-ab*' };
- term vectors for the 3-grams = [13.98, 12.75, 13.34, 12.87];
- and a TF-IDF vector, setting elements to the TF-IDF value of the corresponding *n*-gram or 0 if the corresponding *n*-gram do not appear in the string.

This process can cause the following two problems. First, the number of all possible *n*-grams increases exponentially as *n* increases. For example, if we applies Beskow's work to a set of 38 characters as the set, hence the number of all possible bi, 3 and 4-grams is 1,444; 54,872; and 2,085,136; respectively. Second, the array for a *n*-gram sub-string is sparse since user ID strings are relatively short. Suppose an ID string includes 20 characters, then 3-grams for 18 elements length arrays, i.e., only 18 among 54,872 elements have values. Similarly, only 17 of 2,085,136 elements have values for 4-gram arrays.

Therefore, using *n*-gram based TF-IDF values as a feature can leads to exponentially increasing feature dimensionality and sparse arrays with increasing *n*. Scalability also becomes problematic, with larger *n* requiring considerably more computing resources, including Central Processing Unit (CPU) time, memory, training time, etc.

B. RARE BUT NORMAL ID

ID detection methods with *n*-gram and TF-IDF assume that rare strings are generated by generation algorithms. However, a normal user ID still can be composed of rare strings (e.g. emoji from combinations of numbers and symbols). Such user IDs include strings which do not appear in many other user IDs and popularity based approaches could detect these normal user IDs as randomly generated. This is why extra features are required in order to capture other properties beside the popularity.

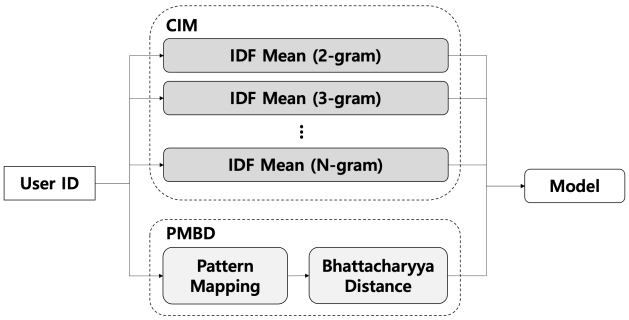


FIGURE 2. CIMP Model Structure.

IV. PROPOSED METHOD

This paper proposes Cumulative IDF Mean with Pattern mapped Bhattacharyya distance (CIMP), an efficient and accurate ID detection method using cumulative IDF mean (CIM) and pattern mapped Bhattacharyya distance (PMBD). The included techniques tackle the curse of dimensionality and rare normal IDs, respectively. CIM reduces dimensionality using mean rather than raw values to represent ID string randomness while reducing dimensionality; whereas PMBD measures pattern randomness in ID strings. The Bhattacharyya distance [4] represents how similar two different distributions are. Patterns extracted from the ID string, and the pattern distribution compared with a uniform distribution by Bhattacharyya distance to measure pattern randomness.

Figure 2 shows the structure of CIMP model. *n*-grams are extracted (*n*=2,3,4) from the user ID string and IDF mean is calculated from IDF values for the extracted *n*-grams. Section IV-A2 discusses why IDF is used instead of TF-IDF. The user ID is converted into patterns and the Bhattacharyya distance from a random distribution calculated.

Following two subsections describes the proposed two techniques in detail and explain how the techniques tackle the problems: the curse of dimensionality and the rare normal IDs, respectively.

A. LIFT OF THE CURSE OF DIMENSIONALITY

Broadly, TF-IDF measures a word s (term s) importance to a document in a collection of documents and *n*-gram extracts fixed-length consecutive sub-sequences from a given string. Both have been widely used in text mining and text-related security applications. As discussed in III-A, *n*-gram based TF-IDF suffers from the curse of dimensionality as the dimension increases exponentially by *n*. We propose a method to compress the size of feature dimensions by using statistical features instead of raw TF-IDF values of extracted *n*-grams. This will reduce the size of required memory and computational cost, while improving the detection performance.

1) TF-IDF

Figure 3 shows an example user ID "*abc-abc*", where we used the entire user ID set to compute TF-IDF values for this

(a) String	"abc-abc"
(b) 3-gram	["abc", "bc-", "c-a", "-ab"]
(c) 3-gram TF-IDF	[13.98, 12.75, 13.34, 12.87]
(d) TF-IDF Mean	13.235

FIGURE 3. Feature candidates and examples (a) user ID string, (b) 3-grams for the string, (c) TF-IDF values for the 3-grams, and (d) mean TF-IDF (d) is the TF-IDF vector of the string.

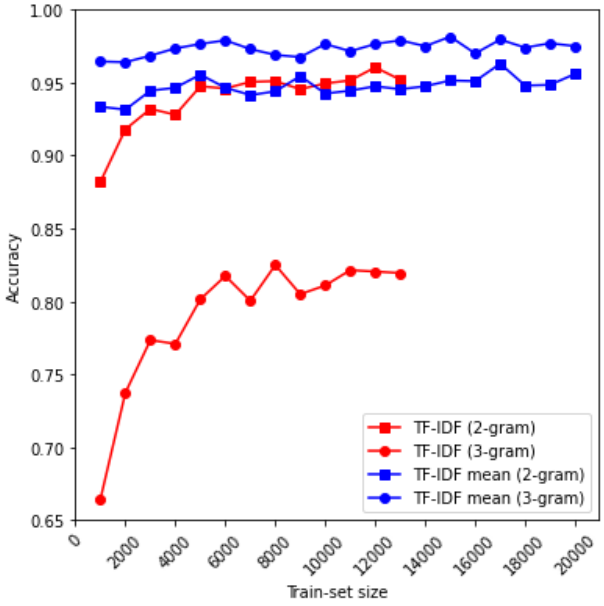


FIGURE 4. Accuracy Comparison between TF-IDF versus TF-IDF Mean.

preliminary analysis. The TD-IDF vector size is the number of possible *n*-grams where each vector element represents the TF-IDF value for the corresponding *n*-gram. The vector is initialized to zero and only relevant elements have TF-IDF values for the specific *n*-grams in the string. A TF-IDF vector is generated for each sample and Beskow's work [22] used the vectors as features.

However, as discussed above, the TF-IDF vector size increases exponentially with increasing *n*, lead to the curse of dimensionality and the vector will be sparse since any given user ID will only include a few *n*-grams. Hence larger *n* cases achieved lower accuracy [22]. Therefore, we propose using statistical TF-IDF measures rather than the raw TF-IDF vector to represent all possible *n*-grams. For example, TF-IDF mean is the mean over all TF-IDF values for the *n*-grams in a user ID.

Figure 4 shows that the method with TF-IDF mean has better accuracy than the method with TF-IDF vector. Experiments were performed with 2-grams and 3-grams, and the accuracy was measured as the size of training dataset was changed from 1,000 to 20,000. Experimental results of 3-gram TF-IDF vector showed only up to 16,000 because memory overheads were too heavy to finish the experiments.

Figure 4 compares TF-IDF vectors and TF-IDF mean for detection accuracy, confirming that TF-IDF mean achieves significantly better accuracy than TF-IDF vectors. Experiments were performed with 2 and 3-grams, and accuracy was measured as the training dataset size increased from 1,000 to 20,000. Experimental results for 3-gram TF-IDF vectors only shows up to 16,000 cases because memory overheads were too heavy to finish the experiment. TF-IDF means provided similar accuracy to TD-IDF for 2 and 3-gram cases with training dataset size $< 3,000$; but 3-gram TF-IDF means provided considerably improved accuracy when training dataset size $> 4,000$. In contrast, the TF-IDF vectors approach achieves superior accuracy using 2-grams than 3-grams.

This improved accuracy confirms the feasibility for n-gram TF-IDF means to distinguish between generated and normal IDs. Thus, Section IV-A1 uses TF-IDF mean to avoid the curse of dimensionality.

2) TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY AND INVERSE DOCUMENT FREQUENCY COMPARISON

TF-IDF was originally designed to statistically measure a words (terms) importance to a document in a collection of documents. A word is considered to be more important than other words if it appears in the document more often than the other words, but if the word often appears in other documents then it is considered to be less important since it has less information to distinguish between the documents. However, the proposed CIMP approach uses TF-IDF to measure n-gram randomness rather than importance. Word or term frequency is a positive factor for importance, but it is not yet clear what affect it may have for randomness. We compare TF-IDF and IDF (excluding TF) performance in terms of detection accuracy, computing TF-IDF mean and IDF mean from the same samples.

Figure 5 shows that IDF mean outperforms TF-IDF mean for user ID detection, verifying that term frequency (TF) is less important (or poor) information for measuring randomness. Therefore, we only use IDF for the remainder of this paper.

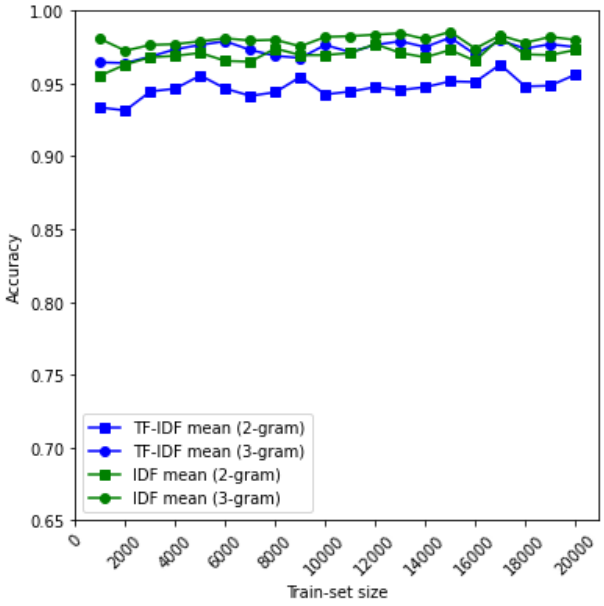


FIGURE 5. Accuracy comparison between TF-IDF mean versus IDF mean.

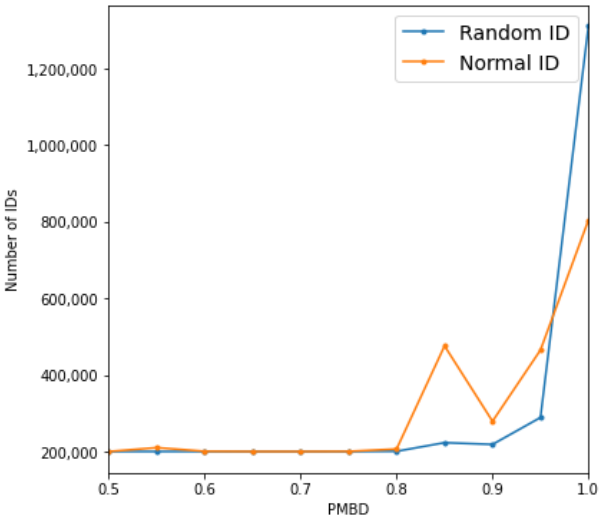


FIGURE 6. Pattern mapped Bhattacharrya distribution.

B. PATTERN MAPPED BHATTACHARYYA DISTANCE

We also propose PMBD to reduce false positives from corner cases (see Section III-B). Since a random user ID generator chooses each character randomly, character distribution will be similar to uniform, i.e., the same occurrence probability for every character. This makes it difficult to compare individual character occurrences. Therefore, we proposed a mapping method to convert each character into its type: alphabet, digit and symbol.

For example, the experiments considered here let user ID be generated from 26 alphabet, 10 digit, and 2 symbol characters. Hence occurrence probabilities = 26/38, 10/38, and 2/38 respectively; assuming a generator creates the user ID randomly. The proposed method calculates the Bhattacharrya distance between the random distribution and measured user ID distribution,

$$\sum_{x \in X} \sqrt{p(x) \times q(x)} \tag{1}$$

where <define x, X, p(x) and q(x)>; $PMDB \in 0, 1$; and smaller $PMDB$ means the distributions are less similar.

Figure 6 shows the PMBD distribution for benign (orange) and randomly generated (blue) user IDs. Randomly generated user IDs have higher PMBD, with mean $PMBD = 0.97$. In contrast, although many benign user IDs also have large PMBD, a significant number of benign user IDs occur with $0.8 \leq PMBD \leq 0.85$. For example the benign user ID "-_-_-_-vv0" looks like a randomly generated ID but it contains

text emojis. This user ID has 3-gram mean $IDF = 11.32$ and $PMBD = 0.72$. If we use just mean IDF alone, this user ID would be classified as a randomly generated user ID. However, including PMBD can successfully classify the user IDs as benign. PMBD can also provide a stand-alone classifier for randomly generated user IDs, which have very high $PMBD$.

C. PROPOSED CIMP MODEL

The CIMP model adds PMBD features to the CIM model as extra features for detecting the corner cases of normal IDs which might be detected as by the CIM model. The CIM and PMBD try to identify its rareness of user IDs based on different aspects. The CIM is based on the assumption that sub-strings of IDs are not popular in other IDs as many IDs are randomly generated. PMBD is based on another assumption that the distribution of each item in user ID string will be different between normal and IDs. By comparing the distribution of an user ID with the expected distribution of randomly generated IDs, a distance can be computed and used as a measure of showing its randomness of an user ID. The CIMP model captures the randomness of user IDs in two different perspectives which complement each other for classifying normal and IDs.

1) CUMULATIVE IDF MEAN MODEL

By using IDF Mean we can use larger n by solving the curse of dimensionality, and as the range of usable n is expanded, it is also possible to use various n-based IDF Mean features in model construction. We designed a model structure that cumulatively expands the range of n that used for IDF Mean to test the effect of the range of n against accuracy. We propose the CIM to computes multiple IDF means for n-gram extraction where $2 < n < N$ to evaluate how the range of n effect the detection performance. For example, if we set $N = 4$, the CIM model computes three IDF means for 2, 3, and 4-grams respectively. Hence we called the proposed model the cumulative model. The evaluation results will be described in Section V.

2) CUMULATIVE IDF MEAN MODEL WITH PMBD

The CIMP model adds PMBD features to the CIM model for detecting normal ID corner cases that might be detected as random IDs by the CIM model. The CIM and PMBD try to identify user ID randomness based on different aspects. CIM assumes that less popular ID sub-strings are randomly generated; whereas PMBD assumes the distribution for each item in a user ID string will differ between normal and random IDs. Thus, a distance can be computed to compare normal and randomly generated ID distributions, and hence identify random user IDs. Thus, the proposed CIMP model captures user ID randomness from complementary perspectives to classifying normal and random IDs.

V. EVALUATION

This section evaluates the proposed CIMP model with real-world data. We first describe the settings and datasets for two experiments to compare ID detection performance between the proposed model and Beskow's work [22].

A. DATASETS AND EVALUATION SETTINGS

In our evaluation, we used three raw datasets as described below.

- (A) Real normal ID set: 3 million IDs of real normal users;
- (B) Randomly generated ID set: 3 million IDs that are generated randomly by our ID generation tool;
- (C) Captured illegal ID set: 8,541 IDs that were used in illegal activities at a real system.

(A) Real normal ID set and (C) Captured illegal ID set are real-world user datasets, obtained from a commercial web service, i.e. Naver Webtoons .[1] (B) Randomly generated ID set is a generated dataset using our random string generation algorithm. With these raw datasets, we made three datasets that are disjoint:

1) Train set: A dataset sampled randomly from (A) and (B), each 10,000 for model training.
2) Artificial evaluation set: A dataset sampled randomly from (A) and (B), each 10,000 for evaluation.
3) Captured evaluation set: A dataset of 8,541 random samples from (A) and 8,541 total from (C) for evaluation.

In the first evaluation, the real normal ID set and the randomly generated ID set are used. Randomly selected 10,000 samples from each set construct a training set of 20,000 samples. Another randomly selected 10,000 samples from the rest of each set construct a test set of 20,000 samples. In the second evaluation, we construct another test set of 17,082 samples: 8,541 captured illegal IDs and randomly selected 8,541 real normal IDs. Note that we use the entire 3 million samples of the real normal ID set for computing IDF values and we implemented the Beskow's work to compare with our proposed models.

B. EVALUATION WITH ARTIFICIAL DATA

In this section, we evaluate our proposed models with the randomly generated IDs to find the optimized value of n and to compare our proposed models with Beskow's work.

1) CUMULATIVE IDF MEAN MODEL

In order to find the optimized value of n, we compare the detection performance of our CIM model with different values of n.

Figure 7 shows receiver operating characteristic (ROC) curves for the CIM model where $2 < n < 6$. CIM(4) achieved highest area under the ROC curve, AUC = 0.9969, reducing for $n \geq 5$. CIM(6) exhibits steep reduction due to overfitting. Since the n increases, the number of n-gram substrings appearing in the entire user ID set decreases. In our case, when n reached 6, the number of appearances of the

[1] https://webtoonscorp.com/en/

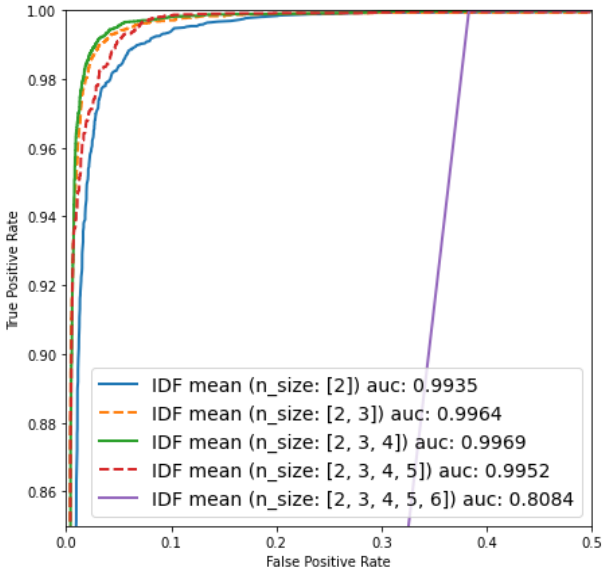


FIGURE 7. ROC curves of Cumulative IDF Mean Models.

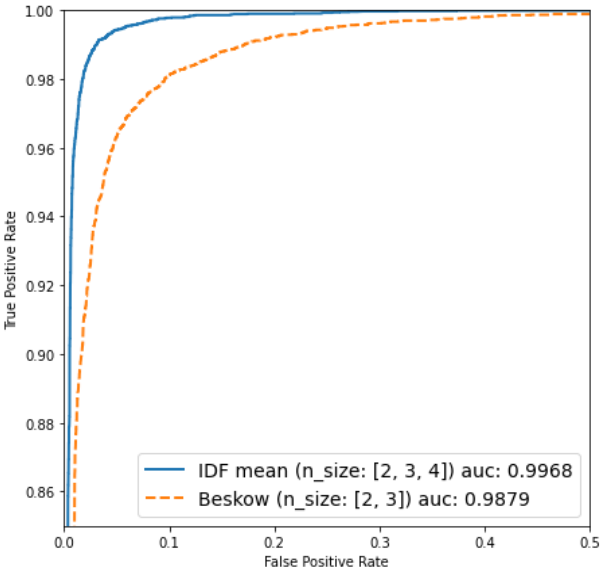


FIGURE 8. ROC curves of Cumulative IDF Mean and Beskow's Model.

TABLE 1. Detection performance comparison.

	Beskow	CIM(4)	CIMP(4)
Accuracy	0.9490	0.9812	0.9829
Error Rate	0.0510	0.0188	0.0171
Precision	0.927	0.9765	0.9789
Recall	0.9747	0.9862	0.987
F1-Score	0.9503	0.9813	0.9829

corresponding substring converges to 0 or close to 1, which leads to overfitting as the feature directly reflects whether the user ID is normal.

This experiment shows that our CIM model can achieve a better detection performance when we use different lengths of n-grams together. However, the detection performance starts decreasing at some point so we need to optimize the value of n for the best performance. The best value of n can be different in different settings but the value of the best n was 4 in our setting.

Now, we compare our CIM model with Beskow's work with the optimized value of n, which is 4. Beskow's work also uses the best setting, TF-IDF values of 2-grams and 3 grams. Figure 8 shows the ROC curves of two models. Our CIM model outperforms Beskow's work. The AUC of two models were 0.9969 and 0.9879 respectively.

2) CUMULATIVE IDF MEAN WITH PMBD (CIMP) MODEL

Although our CIM model outperforms Beskow's work, we have noticed that our CIM model detects rare normal IDs as malicious. To reduce these false positives, PMBD is introduced with our CIM model and it constructs our CIMP model.

Figure 9 shows ROC curves for CIM and CIMP models, with $n = 4$ for both two models. It is expected that the CIMP model works better when a detection system requires a low false positive.

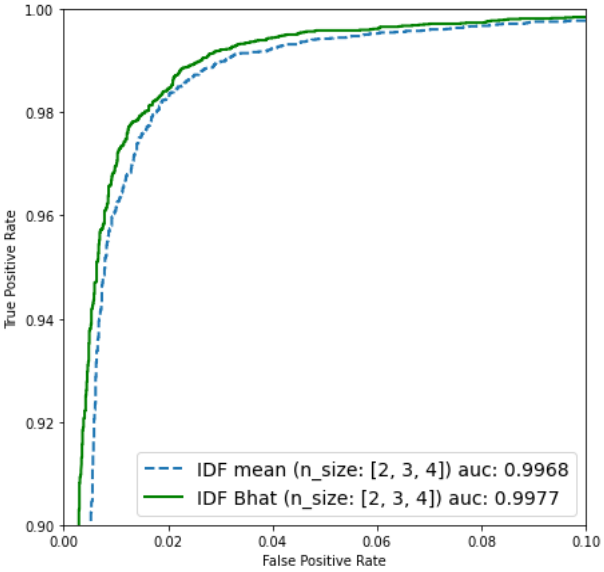


FIGURE 9. PMBD model.

Table 1 shows performance metrics for Beskow, CIM and CIMP. CIM(4) and CIMP(4) achieved the best performance for all metrics, with significantly reduced error rate = 0.0171 compared with Beskow (0.0510). 66% of the errors were reduced by CIMP(4).

C. EVALUATION WITH CAPTURED ILLEGAL ID DATASET

This experiment used the proposed model with a real dataset that containing 8,541 captured illegal IDs, along with 8,541 real normal user IDs, i.e., 17,082 total samples in the dataset. Table 2 that CIMP outperforms CIM for all metrics for this dataset. In particular, CIMP reduced error rate by 33.7%. Thus, PMBD reduces the false positives on rare user ID cases.

TABLE 2. Model performances with real data.

	CIM(4)	CIMP(4)
Accuracy	0.9872	0.9915
Error Rate	0.0128	0.0085
Precision	0.9792	0.9852
Recall	0.9973	0.9980
F1-Score	0.9882	0.9916

VI. CONCLUSION AND FUTURE WORK

Malicious users using fake user accounts put services and normal users at risk through auto-generated random IDs. We investigated detecting fake user accounts through identifying randomly generated IDs. Current conventional methods are insufficient for this application. We optimized the technique using *n*-gram appearance frequency, which has been employed previously, on the basis of *n* compared to ID string length. Subsequently, we proposed CIMP, a new feature extraction method to analyze user ID string pattern and identify random IDs. The proposed CIMP model significantly improved detection, reducing false positive outcomes in particular, and is applicable to various user ID patterns.

As for future work, we intend to explore ways to broaden the scope of fake account detection by integrating static and dynamic features. For example, using user profile informations such as user email address, number of friends and followers or user's behavioral informations such as languages and topics that included in postings which written by each accounts. References [23] [24] [25] can be used to determine whether the account is fake or not.

APPENDIX A

The following pseudo code is implementation of IDF Mean feature. The CIM model consists of IDF features with different n_sizes.

```
class IDFMean:
    def make_ngram_set(self, user_id):
        result = list()
        user_id_len = len(user_id)
        ngram_len = user_id_len - self.n_size + 1

        for i in range(ngram_len):
            n_gram_part = user_id[i:i+self.n_size]
            result.append(n_gram_part)

        return set(result)

    def make_dict(self, user_id):
        ngram_set = self.make_ngram_set(user_id)

        for ngram_part in list(ngram_set):
            if n_gram_part not in self.parts:
                self.parts[n_gram_part] = 0

            self.parts[n_gram_part] += 1
            self.ngram_part_total_count += 1

        return ngram_set

    def gen_dict(self, df_raw_info):
        user_id.apply(lambda x: self.make_dict(x))

        for p in self.parts:
            total_doc_cnt = df_raw_info.shape[0]
            r = total_doc_cnt / (1 + self.parts[p])
            idf = math.log(r)
            self.info_dict[part] = idf

        self.minimum = math.log(total_doc_cnt)

        return self.info_dict

    def preproc(self, user_id):
        ngram_set = self.make_dict(user_id)
        idf_list = list()

        for n in list(ngram_set):
            r = self.info_dict.get(n, self.minimum)
            idf_list.append(r)

        return stat.mean(idf_list)
```

The following pseudo code shows the implementation of PMBD.

```
class PatternBhat(Feature):
    def __init__(self):
        self.info_dict = {
            "ascii": 26/38,
            "digit": 10/38,
            "bar": 2/38
        }

    def preproc(self, user_id):
        def cp(user_id, pattern_list):
            result = 0
            for char in user_id:
                if char in pattern_list:
                    result += 1

            return result

        len_id = len(user_id)
        list_a = list(string.ascii_lowercase)
        list_d = list(string.digits)

        user_id_info = {
            "ascii": cp(user_id, list_a) / len_id,
            "digit": cp(user_id, list_d) / len_id,
            "bar": cp(user_id, ["-", "_"]) / len_id
        }

        result = 0
        for char_type in self.info_dict.keys():
            c_type = user_id_info[char_type]
            occ = self.info_dict[char_type]
            type_root = math.sqrt(c_type * occ)
            result += type_root

        return result
```

REFERENCES

[1] T. Graham and R. Ackland, "Do socialbots dream of popping the filter bubble? The role of socialbots in promoting deliberative democracy in social media," in *Socialbots and Their Friends: Digital Media and the Automation of Sociality*. Oxfordshire, U.K.: Taylor & Francis, 2017.

[2] S. Choi and J. Kwak, "Feature analysis and detection techniques for piracy sites," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 5, pp. 2204–2220, May 2020.

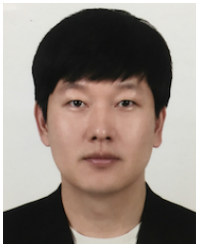
[3] I. Ro, J. S. Han, and E. G. Im, "Detection method for distributed web-crawlers: A long-tail threshold model," *Secur. Commun. Netw.*, vol. 2018, pp. 1–7, Dec. 2018.

[4] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, no. 1, pp. 99–109, 1943.

[5] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, *arXiv:1611.00791*.
[6] B. Yu, D. L. Gray, J. Pan, M. D. Cock, and A. C. A. Nascimento, "Inline DGA detection with deep networks," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, New Orleans, LA, USA, Nov. 2017, pp. 683–692.
[7] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of DGA domain names," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8.
[8] D. S. Berman, "DGA CapsNet: 1D application of capsule networks to DGA detection," *Information*, vol. 10, no. 5, p. 157, Apr. 2019.
[9] Y. Qiao, B. Zhang, W. Zhang, A. K. Sangaiah, and H. Wu, "DGA domain name classification method based on long short-term memory with attention mechanism," *Appl. Sci.*, vol. 9, no. 20, p. 4205, Oct. 2019.
[10] F.-L. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: A survey," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 5, no. 6, pp. 741–760, Nov. 2021.
[11] L. Yang, G. Liu, Y. Dai, J. Wang, and J. Zhai, "Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework," *IEEE Access*, vol. 8, pp. 82876–82889, 2020.
[12] D. Ma, S. Zhang, F. Kong, and Z. Fu, "Malicious domain name detection based on Doc2Vec and hybrid network," in *Proc. 8th Annual Int. Conf. Geo-Spatial Knowl. Intell.*, Xi'an, China, Dec. 2021.
[13] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2010, pp. 48–61.
[14] A. Cucchiarelli, C. Morbidoni, L. Spalazzi, and M. Baldi, "Algorithmically generated malicious domain names detection based on n-grams features," *Expert Syst. Appl.*, vol. 170, May 2021, Art. no. 114551.
[15] M. Antonakakis, R. Perdisci, Y. Nadji, N. V. S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," in *Proc. 21st USENIX Secur. Symp.*, Bellevue, WA, USA, Aug. 2012, pp. 491–506.
[16] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, Egham, U.K., Jul. 2014, pp. 192–211.
[17] J. Selvi, R. J. Rodríguez, and E. Soria-Olivas, "Detection of algorithmically generated malicious domain names using masked N-grams," *Expert Syst. Appl.*, vol. 124, pp. 156–163, Jun. 2019.
[18] C. Xu, J. Shen, and X. Du, "Detection method of domain names generated by DGAs based on semantic representation and deep neural network," *Comput. Secur.*, vol. 85, pp. 77–88, Aug. 2019.
[19] M. Alaeiyan, S. Parsa, and M. Conti, "Detection of algorithmically-generated domains: An adversarial machine learning approach," *Comput. Commun.*, vol. 160, pp. 661–673, Jul. 2020.
[20] D. M. Freeman, "Using Naive Bayes to detect spammy names in social networks," in *Proc. ACM Workshop Artif. Intell. Secur.*, Berlin, Germany, Nov. 2013, pp. 3–12.
[21] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.
[22] D. M. Beskow and K. M. Carley, "Using random string classification to filter and annotate automated accounts," in *Proc. 11th Int. Conf. SBP-BRiMS*, Washington, DC, USA, Jul. 2018, pp. 367–376.
[23] P. K. Roy and S. Chahar, "Fake profile detection on social networking websites: A comprehensive review," *IEEE Trans. Artif. Intell.*, vol. 1, no. 3, pp. 271–285, Dec. 2020.
[24] F. Masood, G. Ammad, A. Almogren, A. Abbas, H. A. Khattak, I. U. Din, M. Guizani, and M. Zuair, "Spammer detection and fake user identification on social networks," *IEEE Access*, vol. 7, pp. 68140–68152, 2019.
[25] E. Van Der Walt and J. Eloff, "Using machine learning to detect fake identities: Bots vs humans," *IEEE Access*, vol. 6, pp. 6540–6549, 2018.

INWOO RO received the B.S. and M.S. degrees in electronics and computer engineering from Hanyang University, South Korea, in 2007 and 2009, respectively. He has been a Researcher with Naver Webtoon Corporation, South Korea, since 2018. His research interests include anomaly detection, user profiling, recommender systems, and time series analysis.

BOOJOONG KANG received the B.S., M.S., and Ph.D. degrees in electronics and computer engineering from Hanyang University, South Korea, in 2007, 2009, and 2013, respectively. He was a Research Fellow at the Queens University of Belfast, from 2014 to 2021. He has been an Assistant Professor with the University of Southampton, since 2021. His research interests include malware analysis, intrusion detection, blockchain, and AI for security.



CHOONGHYUN SEO is a Research Engineer with Naver Webtoon Corporation, South Korea with more than 15 years of software engineering experience. His areas of research and recent projects focus on watermarking and prediction of abuse techniques for intellectual property protection of digital content. His research interests include machine learning and image processing technologies.

EUL GYU IM received the B.S. and M.S. degrees from Seoul National University, in 1992 and 1994, respectively, and the Ph.D. degree from the University of Southern California, in 2002. He was with the National Security Research Institute, Daejeon, South Korea. He is a Faculty Member with the Department of Computer Science, Hanyang University, Seoul, South Korea. His research interests include malware analysis, malicious traffic analysis, and the IoT security.