# AI-Based Control Approach of .SN Reserved Domain Names (aIDN.SN)

Evrard Cabrel Nguemeyou Tchouangang(✉), Ahmadou Ndiaye, Bassirou Kassé, Alex Corenthin, and Idrissa Sarr

Université Cheikh Anta Diop de Dakar, Dakar, Senegal
{evrard.nguemeyoutchouangang,ahmadou14.ndiaye,bassirou.kasse, alex.corenthin,idrissa.sarr}@ucad.edu.sn
https://fst.ucad.sn/

**Abstract.** DNS (*Domain Name System*) is the distributed computer service that associates Internet domain names with their IP addresses. For the sake of business or malicious actions, some persons can make various DNS abuses such as squatting, typosquatting, and so on. To prevent such abuses, DNS managers elaborate a couple of policies and tools to double-check whether a domain name is compliant or not. Some existing solutions rely on identifying a list of reserved terms and proceed to syntactic verification before allowing the record of a new domain name. Such an approach, unfortunately, does not prevent typosquatting and Soundsquatting. To overcome such a drawback, we introduce a control approach made of both syntactic and phonetic verification supported by a classification module for decision-making. Our approach is validated over a set of 9726 domain names and around 5200 reserved terms. Results show the effectiveness of our approach and how it overcomes the existing algorithms devised for terms-reserved compliance check.

**Keywords:** DNS · DNS abuses · Intelligent domain names systems · squatting · typosquatting · cybersquatting · Soundsquatting

## 1 Introduction

DNS (*Domain Name System*) is the distributed computer service that associates Internet domain names with their IP addresses. Domains are organized in a hierarchical structure where the top is called root and represented by a dot. The domains immediately below the root are called Top Level Domains (TLDs). To simplify, TLD can be categorized as follow: Country Code Top-Level Domain (ccTLD), consisting of two letters identifying an independent country or territory (*e.g.* sn for Senegal, cm for Cameroon, fr for France); generic Top-Level Domain (gTLD), consisting of three or more letters generally identifying the sector of activity in which the individuals or organisations using them operate (.com, .org, .info). ccTLDs are managed by countries. For instance, NIC Senegal is the


K. Cheikh M. F. Kebe et al. (Eds.): InterSol 2024, LNICST 610, pp. 275–286, 2025.
https://doi.org/10.1007/978-3-031-86493-3_22

registry operator of .sn. To fulfill its mission, NIC Senegal has defined a naming policy and a set of services. Compliance with the legal policy is checked using the domain name management platform, which enables *a priori* checks to be carried out to ensure that created domain names cope with the naming framework. Additional checks can also be performed *retrospectively* for specific needs.

However, the existing control system suffers from two drawbacks. Firstly, a priori control is carried out based on a set of reserved terms that refer either to prohibited names (terms with sexual connotations or contrary to public decency, terms that may offend religious or gender sensitivities, etc.) or to names subject to conditions. Thus, the efficiency of the control depends on the consistency of the reserved terms list, which is currently updated manually and not regularly. As a result, some words can violate the legal framework sometimes without being detected even during subsequent checks. Secondly, the a priori control algorithm is based on a strict syntactic similarity. Then, only domain names that are literally identical to those already listed in the database of reserved terms are rejected. As a result, some person with malicious intent, commonly known as "abusers", may use terms that are syntactically different but phonetically identical. This type of abuse, more known as "squatting", consists of registering a domain name similar to another in order to profit from the latter. Unfortunately, this type of abuse is more difficult to control in advance. For example, some deep checking have revealed that certain people had registered names such as **fatik.sn** or **fatic.sn** to pass the filter set on **fatick.sn**, which is a city in Senegal and should only be registered as a domain name by the mayor's office. Even though it is possible to detect such squatting abuses, it is important to point out that they are often costly in terms of time and financial resources.

In addition, artificial intelligence is more and more used to face abuses on Internet and to optimise the management of registry services. For instance, one can rely on work done in [1,2] which generates domain names to update the database of reserved terms. Additional works [3,4] have used artificial intelligence to detect domains supplied by DGAs (Domain Generation Algorithm). Unfortunately, there is no one-size-fit-all solution that can be used to face all abuses. This is more true with the variety of sociolinguistic context and the fact that numerous African languages are poorly represented in AI solutions.

The aim of this article is to extend existing AI solution to address the problem of legal framework control in Senegal. To this end, we start from the assumption that the guarantee of both syntactic and phonetic difference is sufficient to semantically dissociate two names and, in turn, will enhance the control squatting-type abuses. We plan to use similarity metrics based on phonetics and domain name classification to pre-analyse compliance and determine what action to take. Due to this classification, creation requests can be automatically rejected or subjected to further checks.

The remainder of this document is divided into five sections. In the second section, we present related works on DNS and abuse management. In the third section, we present the problem statement, i.e. the details of the problem to be solved. In the fourth section, we present our approach, which includes lan-

guage detection, phonetisation and domain classification. In the fifth section, we present the evaluation of the results obtained before concluding and presenting our research perspectives in the sixth section.

## 2 Related Work

Squatting are abuses linked to domain name registration and consist of registering a domain name similar to another in order to take advantage of the latter. In the remainder of this section, we address cybersquatting abuses [5,6] while focusing more specifically on typosquatting abuses [7] and DNS response hijacking such as DNS spoofing [8,9]. We present these works according to its evolution since the 90s.

From 1990 to 2000, the problems of domain name squatting emerged without any real solution. G. Andrew Barger proposed a hierarchical model of the registration and Internet architecture for domain names in order to resolve disputes of this type [10]. With no further solutions, in 1998 ICANN was created to intervene in the resolution of domain name disputes. Then, in 1999, the UDRP (Uniform Domain Name Dispute Resolution Policy) was adopted to effectively resolve disputes and also manage trademark rights violations.

From 2001 to 2010, the abuses did not change much, they just intensified, claiming more victims. In 2001, a new paradigm for intellectual property was defined by Susan Thomas Johnson. At first, several solutions were proposed to counter these abuses, such as the use of the federal trademark registration procedure proposed by John [11], to create a broader and fairer solution, and the application of the Trademark Act 194 of 1993 and/or the Unlawful Competition Act to cases of cybersquatting [12,13]. Later, around 2009, players began to look for solutions that would make it possible to prevent these abuses [14].

From 2011 to 2023, there was a big wave of interest in resolving DNS abuse. We have the soundsquatting approaches, which are approaches based on the pronunciation of the domain name. Nikiforakis et al. [15] use their approach to generate and deploy a series of domains to test their effectiveness. In the same vein we have the works [16,17] which propose approaches based on artificial intelligence to generate possible squatting domains. We also have work that, by analysing the recurring typing errors of Internet users, manages to generate possible malicious domains. In addition to this work, we can add the work done in relation to DGA (Domain Generation Algorithm). These involve using methods such as N-grammes [18], machine learning [19] or, more specifically, neural networks [20] to detect the domains produced by DGA algorithms.

Existing studies show the relevance of DNS abuse problem, however, we have noticed that they are usually based on the generation of abusive domain names and associate it with a syntax check. This may raise a problem since abusive domain name is sometimes different or even very close semantically to those generated, then they would not be able to be detected. To overcome this, our approach uses similarity metrics based on phonetics to prevent all words with strong similarity to the prohibited ones.

## 3 Problem Statement

As stated in the introduction, the main problem of our study is to check the compliance of domain names using a more intelligent strategy in order to better deal with abuse. In addition, we recall that verification of the legal framework conformity is carried out using a set of rules. These can be summarised and formalised as follows.

Let $N$ be the set of all domain names already registered, $N = \{n_1, ..., n_n\}$. Given $R$, the list of all reserved terms in such a way that $R = \{r_1, .., r_m\}$, we define $Gen(r_i) = R - \{r_i\}$ the function that lifts the restriction on $r_i$. A new domain name $(n_i)$ can be registered if one of the following two rules is met:

- **Rule 1:** $\forall r \in R, \forall n \in N, n_i \neq r \wedge n_i \neq n$.
- **Rule 2:** $\exists r \in R \quad | \quad r = n_i, \forall n \in N, n_i \neq n \wedge Gen(r)$.

In other words, a name is added if it is neither registered nor listed in the reserved terms. If the name is on the reserved list, generating a code to lift the restriction is a necessary condition for registration. Unfortunately, this approach only blocks the registration if a new unregistered domain name $n_i$ is syntactically identical to a reserved term $r_i$, *i.e.*, if $Sim(n_i, r_i) = 0$.

As a consequence, one might find a domain name $n'_i \quad | \quad Sim(n'_i, r_i) \leq \epsilon \wedge \epsilon \approx 0$. This situation reflects cases where someone records a name by removing a silent character or a character that does not change its phonetisation or semantics when pronounced. Therefore, problem statement is twofold. To specify them, we start from the assumption that the guarantee of a double syntactic and phonetic difference is sufficient to semantically dissociate two names and, in turn, will enhance the control of squatting abuses.

1. Let $\Gamma$ be the set of domain names eligible for registration and $\bar{x}$ the phoneme of the name $x$, i.e. the phonetic pronunciation of $x$. The first question addressed is to know to what extent $x$ should be added to $\Gamma$ in order to guarantee that for any $x \in \Gamma, \quad \bar{x}$ is enough distant from any $\bar{r}, \forall r \in R$.
2. The second question is how to ensure that any domain name $x$ added to $N$ through the answer to the previous question is not removed subsequently for the same reasons we were trying to avoid.

## 4 Proposed Solution

We propose a mechanism that detects any new domain name close to reserved terms. The main goal is to leverage the existing control algorithm that relies entirely on verifying syntactically the new domain name with reserved terms. By doing so, we aim to verify the eligibility of new domain name request by integrating both syntaxic and phonetic controls. This will avoid typosquatting abuses by ensuring that any new domain is different syntactically and phonetically with existing domain names. To this end, we define a similarity function based on the phonetics of words. Basically, two words are considered as close if their phonetics are quite similar. Moreover, our solution is made of two main building blocks: a domain eligibility checker and a domain name classifier.

### 4.1 Domain Name Eligibility Check (EligCheck)

Eligibility is checked in two phases: a preparation phase, which consists of phonetising the name, followed by a measurement of its similarity to the reserved terms.

**Phonetization of Domain Names.** As a reminder, phonetization, or the act of phonetizing, consists in basing the spelling of a word, text or language on its pronunciation. Graphical representation of a word is called a phoneme. In general, it involves the correspondence between the graphemes (writing units) and phonemes (sound units) of a language. To choose a phonetization tool, we found several solutions in the literature, three of which caught our eye after certain conditions had been met. The choice of solutions to be used is based on a comparative study and is presented in Subsect. 5.2. It should also be noted that the phonetisation process varies according to the languages and systems used. This is why we precede phonetisation with language detection. For this purpose, we assume that every domain name is either in French or in English. By doing this, we assume that even Wolof words, written in the Latin alphabet, can be read in French. We plan to test this hypothesis in future work. To carry out language detection, we rely on existing tools. In the Subsect. 5.1, we make a comparative study of these tools in order to choose the most appropriate in our context.

**Similarity Calculation.** Once the word has been phonetised, we calculate its similarity to the reserved terms. In fact, calculating the similarity between two strings of characters involves measuring how similar or close they are to each other. In machine learning, one of the simplest and most popular distances [21,22] is levenshtein's distance [23]. This will be used in this work to calculate the similarity between two domain names. It is defined by the following formula.

$$lev(a,b) = \begin{cases} max(|a|,|b|) & : \text{if } min(|a|,|b|) = 0 \\ lev(a-1,b-1) & : \text{if } a[0] = b[0] \\ 1 + min \begin{cases} lev(a-1,b) \\ lev(a,b-1) \\ lev(a-1,b-1) \end{cases} & : \text{else-if} \end{cases} \quad (1)$$

In the formula 1, for two domain names $a$ and $b$, the levenshtein distance *lev(a,b)* is the measure of the difference between two domain names. In other words, the minimum number of characters to be inserted, replaced or deleted to move from one domain name to another. This distance is defined with the two strings a and b; $|a|$ the cardinal of a (or its number of letters); and $a-1$ the string a truncated by its first letter i.e. $a[0]$.

**Identify the Eligibility of a Domain Name.** To determine whether a new domain name, $x$, is eligible, we denote by $\bar{x}$ the phonetic of $x$ and $Sim(\bar{x},\bar{r})$, the similarity function used to calculate the distance separating $\bar{x}$ from the reserved term $\bar{r}$ the phonetics of x and r respectively. We declare $x$ to be eligible if:

- $x \notin N$ and
- $\forall r \in R, \quad Sim(\bar{x}, \bar{r}) \in [0,1]$ avec $\quad Sim(\bar{x}, \bar{r}) = lev(\bar{x}, \bar{r})/max(|x|, |r|)$

In other words, a name is eligible if it is not already registered and its phoneme is not similar to any word reserved up to a threshold.

### 4.2 Domain Name Classifier (DNClass)

This is the second component of our approach. It implements a variant of the KNN Machine Learning algorithm with k = 1. Actually, the relative algorithm evaluates the distance between the phonetic of any new domain name to the phonetics of all reserved domain names. Based on the result, the new domain is inserted in one of the predefined classes. In fact, theses classes can represent predefined actions to run in order to ensure that any new domain name is complaint with the registry's policies. Based on the Senegalese registry framework, we have identified three classes in which any new domain name can be placed. Let us assume $\Gamma$ being the set of eligible domain names and $\alpha$ and $\beta$ as two variables. Then the three classes are represented by $\Gamma_\alpha$, $\Gamma_{\alpha\beta}$ and $\Gamma_\beta$ with $0 \leq \alpha < \beta \leq 1$. We define these three classes as follows:

- **Class 0** ($\Gamma_\beta$): This is the class representing eligible domain names that must be registered directly. This class includes all domain names that comply with the registry's policy. $\forall \gamma \in \Gamma$, $c \in \Gamma_\beta$ si $\forall r \in R$, $Sim(\bar{\gamma}, \bar{r}) \in ]\beta, 1]$.
- **Class 1** ($\Gamma_{\alpha\beta}$): This is the class of domain names requiring a second check by another filter before validation. $\forall \gamma \in \Gamma$, $\gamma \in \Gamma_{\alpha\beta}$, si $\exists r \in R$, such that $Sim(\bar{\gamma}, \bar{r}) \in ]\alpha, \beta]$.
- **Class 2** ($\Gamma_\alpha$): This is the class of domain names that must be blocked. This class represents domain names that are very close phonetically to the reserved domain names. $\forall \gamma \in \Gamma$, $\gamma \in \Gamma_\alpha$, si $\exists r \in R$, such that $Sim(\bar{\gamma}, \bar{r}) \in [0, \alpha]$.

One can notice that the numbers of classes can vary from one registry to another based on the policies. Bearing this in mind, the numbers of variables is set accordingly to numbers of classes that depicts mostly the implemented registry policies. In other words, our solution can be considered enough generic to be applied easily on other TLD registries.

## 5 Evaluation of the Proposed Solution

In this section, we conduct an evaluation of our solution to demonstrate its effectiveness. We began by conducting a comparative study of language detection and phoneticization solutions to select the most suitable one for our purposes. Following the comparative studies, we proceeded to evaluate the solution. For the latter, we used the two complete databases for the .SN TLD: the registered domain names database, containing 9726 domain names, and the reserved terms database, containing 4097 domain names. These two databases have 201 domain names in common, corresponding to reserved domain names with registration authorizations.

### 5.1 Language Detection Tools

There are various language detection tools based on AI process that can be ranged from natural language processing algorithms (Googletrans, FastText) to probabilistic statistics (Langdetect), linguistic features (Langid) or linguistic dictionaries (Dictionnaire CMU). We can have tools that are not intended for basic language identification but can be used as such. This is the case with Eng_to_ipa, whose basic purpose is English phonetisation, but which will be used here as a language detection tool. To choose a tool, we first rely on free accessibility criteria. Bearing this in mind, we compare the following solutions: CMU [24], Langid [25], Langdetect [26] and the Eng_to_ipa [27] tool. Afterward, we use the two sets of data (13622 domain names with 11186 names in French and 2436 names in English) to assess which tools is best to detect the language. Precisely we evaluate the different tools based on the following metrics: Elements detected per language, true positives (TP), false positives (FP).

$$Accuracy = \frac{TP + TN}{All\_elements} \tag{2}$$

**Table 1.** Test results for selected language detection tools

| Tools | French elements | TP French | FP French | English elements | TP English | FP English |
|---|---|---|---|---|---|---|
| CMU | 11241 | 11189 | 28 | 2381 | 2378 | 3 |
| Eng_to_ipa | 11186 | 11186 | 0 | 2436 | 2430 | 6 |
| Langid | 4985 | 4411 | 574 | 8637 | 1856 | 6780 |
| Langdetect | 12324 | 10123 | 2201 | 1298 | 229 | 1069 |

Table 1 shows the results obtained from the language detection tools on the whole dataset. We can see that Langid and Langdetect tools produce more false positives compared to CMU and Eng_to_ipa tools.

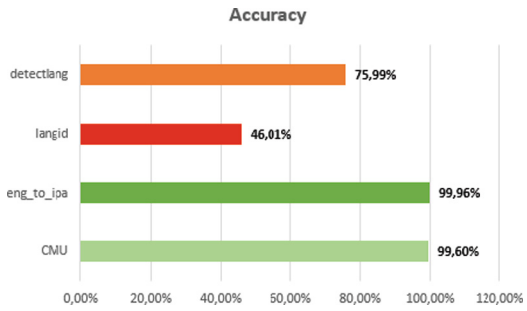


**Fig. 1.** Accuracy of different language detection tools

Figure 1 illustrates the results of the accuracy calculation given by the formula 2. We can see that the eng_to_ipa and CMU tools provide the best results with over 99% accuracy. However, when we add up the number of false positives for the two solutions as shown in Table 1, the CMU method accumulates thirteen-one false positives while the method with eng_to_ipa accumulates six false positives. Finally, to evaluate our solution, we will use the Eng_to_ipa tool for language detection.

### 5.2 Phonetization Tools

As previously, we rely on free tools to consider the following ones: eng_to_ipa [27], Espeak [28], and epitran [29]. Since the phonetisation tool depends tightly to the language, we have Eng_to_ipa tool, which is dedicated to English words, and Espeak and Epitran tools for both languages (English and French). We conducted an empiric evaluation around thousands of domain names for both french and English words. A rigorous analysis had let us know that Epitran and Espeak give best results than others respectively in french and english. Due to lack of space, we skip details of this evaluation.

### 5.3 Overall Solution Evaluation

In this section, we evaluate our overall solution against the existing solution. To achieve this, we will first determine the optimal $\alpha, \beta$ threshold values using an adaptive approach. Once the thresholds have been determined, we'll calculate the gains of our solution compared with the results of the existing solution.

To evaluate the gain, we introduce a fourth class (class 3) which represents the class of reserved domain names that have been registered by lifting the restriction using the $Gen()$ function. This class allows us to correctly evaluate the gain because it represents all the domain names that were blocked by the existing solution. We declare $r$ in class 3 if $r \in R \wedge Gen(r)$. We use the first dataset which represents all registered domain names (9726 domain names). It is divided into four classes. We have 8664 in *class 0 domain names*, 288 in *class 1 domain names*, 573 in *class 2 domain names* and 201 in *class 3 domain names*. We use the Score F1 given by the Table 2 to optimize the $\alpha$ and $\beta$ thresholds.

**Table 2.** Evaluation measures

| Details | Measures | Formula |
|---|---|---|
| **TP:** True Positive<br>**FP:** False Positive | Precision | $\frac{TP}{TP+FP}$ (4) |
| **TN:** True Negative<br>**FN:** False Negative | Recall | $\frac{TP}{TP+FN}$ (5) |
| | *Score F1* | $\frac{2XPrecionXRecall}{Precision+Recall}$ (6) |

- **Precision**: Measures the proportion of positive instances correctly identified among all instances identified as positive by the model.
- **Recall**: Measures the proportion of correctly identified positive instances among all the truly positive instances in the data.
- ***Score F1***: Harmonic mean of precision and recall.

**Optimization of $\alpha$ and $\beta$ Thresholds:** We set up an approach to determine the best pair of thresholds ($\alpha$, $\beta$) to use to evaluate our solution. First, we fix $\alpha$ and vary $\beta$. This step allows us to identify the best value of $\beta$ for which we have the best *Score F1* for class 1 (representing the category of abuse defined by $\beta$). Once this value has been found, we fix $\beta$ at this value and vary $\alpha$ to determine its optimal value. In this way, we obtain the optimal value for the pair ($\alpha$, $\beta$).

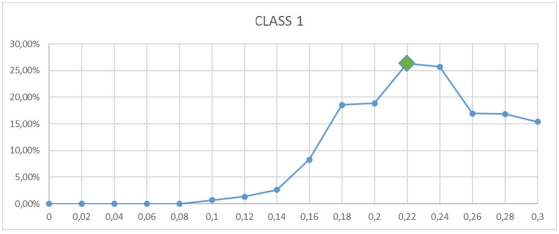


**Fig. 2.** Variation of *Score F1* for class 1 with $\alpha = 0.00$

Figure 2 illustrates the variation of the *Score F1* on all the data for class 1. We set ($\alpha$) to 0.0 and we vary ($\beta$) from 0.0 to 0.30 with a step of 0.02. Here we only look at class 1 because it is the only class directly impacted by the variation of the $\beta$ threshold. Choosing the best value for the $\beta$ threshold means taking the value for which the *Score F1* is highest for class 1. By observing Fig. 2, we note that the value for which the curve is the highest is **0.22** which corresponds to the best value of $\beta$.

For the rest, we set the value of $\beta$ to 0.22 and determine the best value of $\alpha$. Knowing that the value of $\alpha$ must be strictly lower than the value of $\beta$, we vary the value of $\alpha$ from 0.0 to 0.2 with a step of 0.02. In this case, the classes affected are classes 1 and 2. We calculate the value of the *Score F1* for each of the classes and take the average. Figure 3 illustrates the variation of the average *Score F1* between class 1 and 2, in relation to $\alpha$. We can see that the best value for $\alpha$ is 0.10, which gives a final *(0.10, 0.22)* for the pair of ($\alpha$, $\beta$). After that,

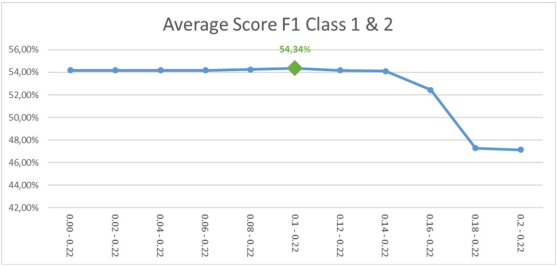


**Fig. 3.** Variation of *Score F1* for classes 1 and 2 with $\beta = 0.22$

**Calculating the Solution Gain:** Once the thresholds have been chosen, our aim is to evaluate the contribution of our solution. This step enables us to assess the effectiveness of our approach by quantifying the gain in misuse detection compared with the existing method. To this end, We use the Gain and *Score F1* given by the formula 3 and the Table 2 to evaluate our solution.

$$Gain(total) = \frac{TP(c1) + TP(c2)}{TP(c3)} \qquad Gain(ci) = \frac{TP(ci)}{TP(c3)} \tag{3}$$

with *TP(ci)* set of true-positives of the class *i*. In order words, the gain is calculating by making the ratio of all True positive of the new solution over the True positive of the existing solution $TP(c3)$. The Table 3 illustrates the calculation of the gain of our solution compared to the existing solution using the formula 3. Looking at the table, we see a gain of *45.27%* for class 1 and *217%* for class 2, totalling an overall gain of *262.69%* in terms of detecting additional abusive domain names. These results clearly indicate that the proposed solution can detect more than twice as many abusive domain names as the existing solution. This significant improvement underlines the increased effectiveness of our model in identifying abuse, positioning our solution as a significantly better performing alternative.

**Table 3.** Evaluation results of the solution

| Details | Measures | Results |
|---|---|---|
| $\alpha = 0.10$ | Class 1 gain | **45.27%** |
| $\beta = 0.22$ | Class 2 Gain | **217.41%** |
| **Total Gain (class 1,2)** | | **262.69%** |

## 6 Conclusion

In this article, we present a solution for managing reserved terms by introducing a dual syntactic and phonetic check. Our approach is based on the calculation

of phonemes and the measurement of the distance between a new domain name and previously defined reserved domains. We used a dataset comprising 9726 domain names, 201 of which were reserved with authorisations. Our approach produced significant results, with gains of around *262.69%* of additional domain names detected as not compliant or under risk of violating the legal framework. However, our solution suffers with two drawbacks which are the time consuming of our algorithm and the lack of automatically integrating new additional reserved terms obtained by the phonetic checking process. The latter drawback has the disadvantage of requiring to run the algorithm indefinitely even though a previous word was already detected as not compliant phonetically with reserved terms rules. Ongoing works are conducted to face such drawbacks.

## References


1. Tahir, R., et al.: It's all in the name: why some URLs are more vulnerable to typosquatting. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 2618–2626. IEEE (2018)
2. Ahmad, I., Parvez, M.A., Iqbal, A.: TypoWriter: a tool to prevent typosquatting. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 423–432. IEEE (2019)
3. Moubayed, A., Aqeeli, E., Shami, A.: Detecting DNS typo-squatting using ensemble-based feature selection & classification models. IEEE Can. J. Electr. Comput. Eng. **44**(4), 456–466 (2021)
4. Moubayed, A., et al.: DNS typo-squatting domain detection: a data analytics & machine learning based approach. In: 2018 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2018)
5. Mercer, J.D.: Cybersquatting: blackmail on the information superhighway. BUJ Sci. Tech. L. **6**, 290 (2000)
6. Bhusari, R.V., Rampure, K.R.: Cybersquatting: a threat to the globalising world. Indian J. Law Legal Res. **3**(2), 2283–2304 (2022). ISBN 2582-8878
7. Moubayed, A., et al.: DNS typo-squatting domain detection: a data analytics & machine learning based approach. In: 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, UAE, pp. 1–7. IEEE, December 2018. ISBN 978-1-5386-4727-1. https://doi.org/10.1109/GLOCOM.2018.8647679. https://ieeexplore.ieee.org/document/8647679/. Accessed 20 July 2023
8. Viegas, E.K., Lopez, M.A., Monteiro, R.M.: Method for detection of DNS spoofing servers using machine learning techniques. Google Patents (2022)
9. Hanley, S.: DNS overview with a discussion of DNS spoofing, November 2000
10. Barger, G.A.: Cybermarks: a proposed hierarchical modeling system of registration and internet architecture for domain names. J. Marshall L. Rev. **29**, 623 (1995)
11. Papavasiliou, J.: Using the federal trademark registration process to create a broader yet fairer solution to domain name contacts. U. Balt. Intell. Prop. LJ **11**, 93 (2002)
12. Ebersohn, G.: Cybersquatting, typosquatting and trade mark law (Part 1). J. S. Afr. Law/Tydskrif vir die Suid-Afrikaanse Reg **2006**(2), 315–329 (2006)
13. Ebersohn, G.: Cybersquatting, typosquatting and trade mark law (Part 2). J. S. Afr. Law/Tydskrif vir die Suid-Afrikaanse Reg **2006**(3), 551–563 (2006)
14. Moore, M.: Cybersquatting: prevention better than cure? Int. J. Law Inf. Technol. **17**(2), 220–231 (2009)

15. Nikiforakis, N., Balduzzi, M., Desmet, L., Piessens, F., Joosen, W.: Soundsquatting: uncovering the use of homophones in domain squatting. In: Chow, S., Camenisch, J., Hui, L., Yiu, S.M. (eds.) ISC 2014. LNCS, vol. 8783, pp. 291–308. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13257-0_17
16. Valentim, R., et al.: AI-based sound-squatting attack made possible. In: IEEE European Symposium on Security and Privacy Workshops (EuroS & PW), pp. 448–453. IEEE (2022)
17. Valentim, R., et al.: Lost in translation: AI-based generator of cross-language sound-squatting. In: 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 513–520. IEEE (2023)
18. Selvi, J., Rodríguez, R.J., Soria-Olivas, E.: Detection of algorithmically generated malicious domain names using masked N-grams. Expert Syst. Appl. **124**, 156–163 (2019)
19. Almashhadani, A.O., et al.: MaldomDetector: a system for detecting algorithmically generated domain names with machine learning. Comput. Secur. **93**, 101787 (2020)
20. Congyuan, X., Shen, J., Xin, D.: Detection method of domain names generated by DGAs based on semantic representation and deep neural network. Comput. Secur. **85**, 77–88 (2019)
21. Quelle distance choisir en machine learning - Pensée Artificielle. https://penseeartificielle.fr/choisir-distance-machine-learning/. Accessed 15 Jan 2024
22. NLP: les techniques et les algorithmes préférés des data scientists- LeMagIT. https://www.lemagit.fr/conseil/NLP-les-techniques-et-les-algorithmes-preferes-des-data-scientists. Accessed 15 Jan 2024
23. Levenshtein, V.I., et al.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Phys. Doklady **10**(8), 707–710 (1966)
24. The CMU Pronouncing Dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Accessed 15 Jan 2024
25. langid on Pypi. en, April 2016. https://libraries.io/pypi/langid. Accessed 11 Jan 2024
26. Shuyo, N.: Language Detection Library for Java (2010). http://code.google.com/p/language-detection/
27. Roberts, R.: English-to-IPA Transcription (2021)
28. eSpeak: Speech Synthesizer. https://espeak.sourceforge.net/index.html. Accessed 09 Jan 2024
29. Mortensen, D.R., Dalmia, S., Littell, P.: Epitran: precision G2P for many languages. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018) (2018)