Sound-skwatter (Did You Mean: Sound-squatter?) AI-powered Generator for Phishing Prevention

Rodolfo Valentim
Politecnico di Torino
Torino, IT
rodolfo.vieira@polito.it

Idilio Drago
Università degli Studi di Torino
Torino, IT
idilio.drago@unito.it

Federico Cerutti
Università degli Studi di Brescia
Brescia, IT
federico.cerutti@unibs.it

Marco Mellia
Politecnico di Torino
Torino, IT
marco.mellia@polito.it

ABSTRACT

Sound-squatting is a phishing attack that tricks users into malicious resources by exploiting similarities in the pronunciation of words. Proactive defense against sound-squatting candidates is complex, and existing solutions rely on manually curated lists of homophones.

We here introduce Sound-skwatter, a multi-language AI-based system that generates sound-squatting candidates for proactive defense. Sound-skwatter relies on an innovative multi-modal combination of Transformers Networks and acoustic models to learn sound similarities. We show that Sound-skwatter can automatically list known homophones and thousands of high-quality candidates. In addition, it covers cross-language sound-squatting, i.e., when the reader and the listener speak different languages, supporting any combination of languages.

We apply Sound-skwatter to network-centric phishing via squatted domain names. We find $\sim 10\%$ of the generated domains exist in the wild, the vast majority unknown to protection solutions. Next, we show attacks on the PyPI package manager, where $\sim 17\%$ of the popular packages have at least one existing candidate.

We believe Sound-skwatter is a crucial asset to mitigate the sound-squatting phenomenon proactively on the Internet. To increase its impact, we publish an online demo and release our models and code as open source.

KEYWORDS

soundsquatting, phishing, machine-learning

1 INTRODUCTION

Cyber-squatting is a phishing attack that relies on the similarities of words to trick users into malicious systems, sites, or content. This type of identity attack has been applied in different contexts, from fake domains [7, 16] and phishing campaigns [22, 23], to hijack the functionality of smart speakers [12, 31] among others. Several cyber-squatting strategies have been hypothesized and demonstrated in practice, including simple/frequent typos [1, 10, 24, 27], visual similarity of words [7] and the collocation of common words [11, 13].

Sound-squatting is a phishing technique that tries to trick users by leveraging words with pronunciation perceptively similar to targets. Sound-squatting is generic and applies to the case of listeners that have to write (or interpret) a word pronounced by another person. Whereas other types of cyber-squatting have been largely studied [25, 30], sound-squatting has received relatively little attention. Most previous studies are limited to the use of lists of known English homophones that lack coverage regarding the scope of such attack technique [16]. Sound-squatting is challenging because it involves languages and multiple aspects that change across people.

Current sound-squatting detection and mitigation strategies rely on manually curated lists of known homophones, i.e., two or more words having the same pronunciation but different meanings, origins, or spelling. For example “new” and “knew”. In such lists, one can enumerate sound-squatting candidates, thus checking whether targets (e.g., domain names, internet profiles, etc.) may be a victim of tentative abuses [16].

This strategy has limitations. First, the set of homophones in a language is limited and it does not consider possibly not-existing words that might have perceptively similar pronunciations, i.e., quasi-homophones. Second, current solutions operate at a word level and do not consider character or syllable replacements and occlusions that have little effect on pronunciation. Third, homophones limit the sound-squatting generation to the same-language case. Being the Internet global, cross-language scenarios are likely popular. Here, the attacker exploits the inability of a person to correctly pronounce or write a word belonging to a foreign language.

We here introduce Sound-skwatter, an AI-based system capable of *automatically* creating sound-squatting candidates. Sound-skwatter generates candidates for any given target name, working at the sub-word level and allowing configurable approximations during the search for candidates. Sound-skwatter is built using a state-of-the-art Transformers Neural Network that can be trained for any language [26]. For training, the network receives as input (i) the International Phoneme Alphabet (IPA) representation of the word and (ii) the spectrogram [21] extracted from the target word pronunciation audio signal. At inference, it recreates the written form (*grapheme*) while also considering pronunciation. Sound-skwatter uses a sequence-to-sequence model to find written alternatives with similar pronunciations.

We first validate Sound-skwatter by comparing the list it generates against lists of known homophones. Here we find that Sound-skwatter can automatically generated 84% of known English homophones, and thousands of additional quasi-homophones, with the model with acoustic feedback that performs better than a simple system that does not include the audio part.

We show how Sound-skwatter can help to mitigate domain name sound-squatting. We instrument it to generate automatically possible sound-squatting candidates of the top popular websites. Astonishingly, about 10% of generated domains exists in the wild, but are unknown to current protection solutions. These include multiple cases we manually identify as malicious and on-sale/parking domains. Interestingly, some of these candidate domains are registered by the legitimate target owner as proactive protection against squatting.

We then check the existence of possible sound-squatting abuse in other domains. We focus on the PyPI repository for Python packages and we look for sound-squatting that target popular packages. Sound-skwatter uncovers multiple cases. While these candidates do not seem malicious, the lack of structure and curation in the PyPI repository – allowing such arbitrary names – is a huge blind spot on the platform.

In sum, our contributions can be summarized as follows:

- We introduce Sound-skwatter, an AI-powered sound-squatting automatic generator.
- We show that Sound-skwatter lays the ground for cross-language sound-squatting search and mitigation.
- We provide a thorough analysis of Sound-skwatter in two contexts, showing that sound-squatting is likely exploited for both domain squatting, and that PyPI packages.

Sound-skwatter can be used for protection against attacks that target, e.g., brands. It results in an inexpensive solution for searching squatting campaigns in less controlled markets, i.e., automatically generating possible domain names that can be or are already abused by attackers.

To allow readers to test Sound-skwatter in practice, we provide an anonymous running demo of the system online at http://54.196.25.136/demo. Sound-skwatter pre-trained models, together with all our source code, will be available freely to the community upon publication.[1]

Next, we provide background information about sound-squatting and the neural network architectures we use in Section 2. We describe Sound-skwatter in Section 3 and validate it in Section 4. We then discuss the results of domain name sound-squatting and PyPI packages sound-squatting in Section 5. We discuss related work in Section 6 before concluding the paper in Section 7.

2 BACKGROUND

2.1 Cyber-squatting and sound-squatting

Cyber-squatting is a class of attack in which a malicious actor tries to impersonate a legitimate resource [4]. Cyber-squatting has gained notoriety with the widespread deployment of domain-squatting, a type of attack in which attackers register fake domain names to divert traffic from popular websites. For example, an in-depth search of over 224 million DNS records in 2018 identified 657 k domain names likely impersonating 702 popular brands [25].

The Mitre Att&ck's CAPEC-631 [5] defines sound-squatting uniquely in the context of domain-squatting, as an attack in which *"an adversary registers a domain name that sounds the same as a trusted domain, but has a different spelling"*. However, sound-squatting is more generic than that. For instance, abuses in Alexa voice assistant have already been identified [12]. In general, the sound-squatting attack tries to diverge users into malicious resources based on the assumption that users (or devices) will confound words with similar pronunciations — e.g., clicking on links, typing in similar malicious words, or misunderstanding an action.

Considering only domain-squatting, the CAPEC-631 also enumerates possible mitigation to sound-squatting: (i) the deployment of additional checks when resolving names in the DNS, together with the authentication of servers, and (ii) the preventive purchase of domains that have potential for sound-squatting. In addition to these protections, the system can warn the user about possibly malicious use of a word.

In all cases, the targets of sound-squatting must know the names attackers will use to impersonate their brands. Sound-skwatter comes precisely to solve this problem, generating candidate names — ranked with their probabilities — that can be used to mitigate the problem proactively.

2.2 International Phoneme Alphabet (IPA)

Continuous speech is perceived in segments. These segments are classified according to how the vocal tract produces them. Each natural language uses a different set of segments. Phonemes are the smallest units in a language to distinguish word meanings. Phonemes are precise phonetic realizations often affected by social variation, regional dialect, etc. The International Phonetic Alphabet is a means to represent such phonemes with standard symbols (IPA) [2].

The International Phonetic Association introduced IPA in the 19th century. IPA uses Latin and Greek script symbols, transcribing sounds (phones). Letters are organized into categories, i.e., vowels and pulmonic/non-pulmonic consonants. Some examples are shown in Table 1 [28].

Table 1 shows that some phonemes may be equivalent to more than one grapheme. This relation varies from language to language, with some languages more *phonetically consistent* than others. Italian and German are two examples of languages with high consistency, i.e., most graphemes have a one-to-one relation with a phoneme. English and many other languages are not phonetically consistent, largely missing such a direct relation. Languages with considerable inconsistency are more prone to confusion when interpreting/writing a pronounced word. Inconsistent languages are thus particularly vulnerable to sound-squatting.

Sound-skwatter uses IPA to encode the input words. There exist solutions that translate any word in the corresponding IPA. For instance the eSpeak NG (Next Generation) TTS engine[2] supports more than 100 languages. Sound-skwatter uses it to derive the input IPA given a target word and language.

[1]Details are omitted to keep authors' anonymity. We suggest accessing the site from an anonymous IP address to avoid unveiling the reviewers.

[2]https://github.com/espeak-ng/espeak-ng

Table 1: Examples of phonemes for American English with the grapheme and some examples (source [28]).

Type	Phoneme	Grapheme	Examples
Consonant	ʃ	sh, ce, s, ci, si, ch, sci, ti	sham, ocean, sure, special, pension
Consonant	v	v, f, ph, ve	vine, of, stephen, five
Vowel	æ	a, ai, au	cat, plaid, laugh
Vowel	eɪ	a, ai, eigh, aigh, ay, er, et, ei, au, ae, ea, ey	bay, maid, weigh, straight, pay

2.3 Transformers Neural Networks

Transformers are a hot topic in the deep neural network community. Initially proposed for translation [26], they have been used as the means to achieve natural language processing (NLP), computer vision (CV), and speech processing.

The vanilla implementation of Transformers consists of an encoder and a decoder, each of which is a stack of N identical blocks. Each encoder includes a MultiHead Attention block and a feed-forward interconnected by *Add and Normalization* layers. The decoder is similar to the encoder; however, it adds to the encoder's architecture a cross-multihead attention mechanism that computes the attention between the input and the previous states of the target.

An important aspect to discuss regarding the Transformers is the difference between the training and the inference. During training, the Transformers use a mask over the target states given as input for the cross-attention head. This mask allows the model to learn multiple states simultaneously.

The inference process happens in steps where the decoder receives the previous states generated to compute the cross-attention. In this way, the output of the next element depends on the sequence of the previously generated elements. This is fundamental to correctly predict the next character in a word or the next word in a sentence.

Sound-skwatter uses Transformers to generate words at the character level. It considers not only the most probable character but also explores possible words that may derive from considering the top-K most probable following characters, generating possible alternative ways to write the same input IPA sequence.

2.4 Mel Spectrogram

The spectrum describes a signal's magnitude and phase characteristics as a function of frequency [20]. A spectrogram is a visual representation of an audio signal decomposed in frequencies where one axis represents time, and the other represents frequencies.

The Mel Spectrogram is a log-scaled version of a Linear Spectrogram. This representation is more suitable for humans because it is easier for our auditory system to distinguish between similar low-frequency sounds than between similar high-frequency sounds. The audio representation in Mel Spectrogram is, therefore, often used in machine learning applications because it helps to focus on the components of frequencies that matter for speech applications. Figure 1 shows the raw audio signal, the Mel Spectrogram, the pronunciation in IPA, and the grapheme for the word *"community."*

Sound-skwatter uses the Mel Spectrogram to learn the representation of a sound and uses the representation to generate graphemes whose pronunciation is close to the original sound. Sound-skwatter also uses the alignment of the pronunciation with the signal to infer

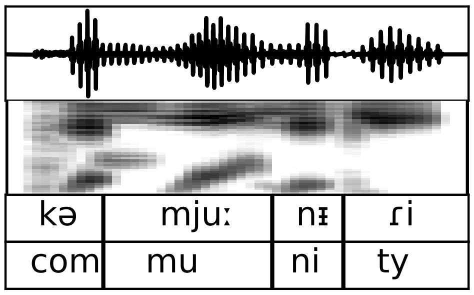


Figure 1: An raw audio wave and its Mel Spectrogram aligned to the pronunciation and grapheme of the word *"community."*

the duration of each phoneme, as shown in Figure 1. Such duration is pivotal also for reconstructing the spectrogram, as discussed in more detail in the following section.

3 SOUND-SKWATTER: SYSTEM DESCRIPTION

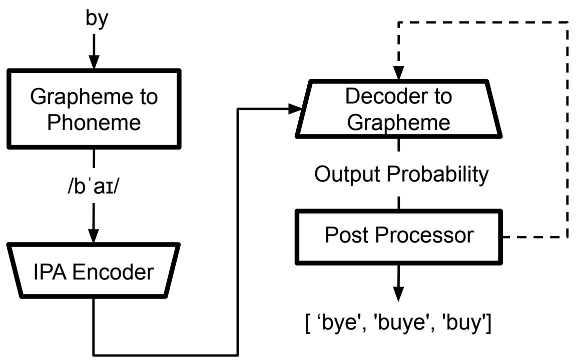
Sound-skwatter is an AI-based system capable of creating sound-squatting candidates that relies on Transformers Neural Network to translate from phonemes to graphemes. We start by describing the part of the architecture used during inference. It operates as depicted in Figure 2a:

(1) `Grapheme-to-Phoneme` — built upon the eSpeak NG (Next Generation) TTS engine[3] — transforms an input word written in grapheme form (e.g., the word `by`) into its IPA representation (e.g., `/bˈaɪ/`).
(2) `IPA-Encoder` then encodes such an IPA word into a vector representation.
(3) `Decoder-to-Grapheme` decodes the vector representation into words written in grapheme form which are quasi-homophones to the input word (e.g., `bye`).
(4) `Post-Processor` selects the quasi-homophones to be presented to the user.

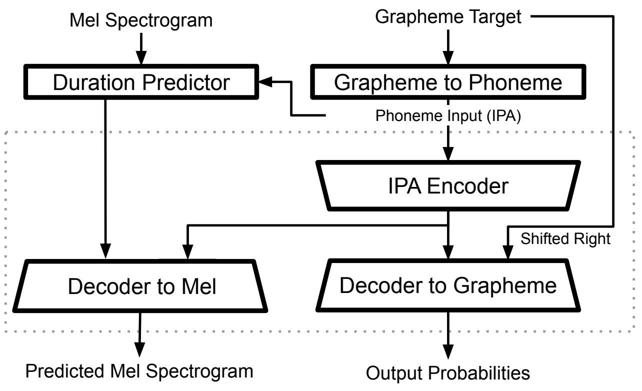
The architecture used to train the `IPA-Encoder` and `Decoder-to-Grapheme` — the core of our proposal — is depicted in Figure 2b. Both modules are trained jointly with one further function, `Decoder-to-Mel`, exploiting multi-modal data coordinated by the `Duration-Predictor` function. The overall training task thus:

- is analogous to a sequence-to-sequence model, where the `IPA-Encoder` and `Decoder-to-Grapheme` together translate from IPA to the target language grapheme. At the

[3]https://github.com/espeak-ng/espeak-ng (visited on 10/10/2022).



(a) Architecture used during inference.



(b) Architecture used during training.

Figure 2: (a) The process to generate candidates is composed of an `IPA-Encoder` that maps the input to a latent space and a `Decoder-to-Grapheme` that reconstructs the input. (b) The training architecture for learning how to generate quasi-homophones. The dotted box highlights the components that are trained for the generation of quasi-homophones: `Duration-Predictor` is a pre-trained function. It receives as input the phoneme translation of a word and the duration of each phoneme in the expected spectrogram and outputs a reconstructed spectrogram and the probabilities that are used to find the grapheme translation.

same time the Mel Spectrogram supervisory signal is reconstructed from the phoneme vector representation and the forecast duration of each phoneme in the spectrogram domain;

- uses as input pairs $\langle x, \texttt{to-Mel}(\texttt{TTS}(x))\rangle$, where x is a word in phoneme format, and `to-Mel` and `TTS` are, respectively, one of the many existing software that generates a Mel Spectogram of audio signals and Text-to-Speech software, eventually first calling the `Grapheme-to-Phoneme` function for having it in IPA. The dual-modality input is required for generating quasi-homophones of good quality;
- outputs the word in grapheme format (via `Decoder-to-Grapheme`) and the Mel Spectogram (via `Decoder-to-Mel`) from the vector representation which `IPA-Encoder` outputs. The loss function used for training needs to carefully balance the training process between these two modalities. The loss is computed by summing the $L1$ and $L2$ Loss of the Mel Spectrogram reconstruction and the Cross-entropy Loss calculated between the predicted and the expected grapheme.

`Duration-Predictor`, a pre-trained function, coordinates the flow of multi-modal data by feeding the predicted temporal duration of each of the phonemes in `Grapheme-to-Phoneme`(x) to `Decoder-to-Mel`. It uses Convolution and LSTM Units to minimize the Connectionist Temporal Classification (CTC) loss [6], i.e., a loss between a continuous time-series— the Mel Spectrogram— and a target sequence— the IPA word. The former is obtained by applying the Fast Fourier transform (FFT) over windowed segments of the audio signal and a transformation to Mel Scale. Given K the length of x, our word in phoneme format, `Duration-Predictor` outputs $\langle d_1, d_2, ..., d_K\rangle$, where d_i represents the number of windowed segments of the audio signal predicted to be associated to the i-th phoneme of x.

Sound-skwatter leverages Transformers Neural Network to train `IPA-Encoder`, `Decoder-to-Grapheme`, and `Decoder-to-Mel` (within the dotted box of Figure 2b). Transformers rely on the self-attention mechanism to learn how to translate a word in IPA back into grapheme format. The output of `IPA-Encoder` is forwarded to `Decoder-to-Grapheme`, which uses the contextual information extracted from the input by `IPA-Encoder`, together with the current state of the output. As discussed in Section 2.3, a Transformer is an auto-regressive model at inference, i.e., it can leverage its own history of predictions to forecast future states.

In particular, `Decoder-to-Grapheme`, given an IPA input and the predictions made for each of the previous $N-1$ characters, forecasts the probabilities of each possible character being the N-th character. It is then the role of `Post-Processor` to look at these probabilities and feed the history back to the `Decoder-to-Grapheme` for the new forecast, as discussed in more detail in the following.

3.1 `Post-Processor`: the Quasi-Homophone Generation

The `Post-Processor` is the last element of the inference: it receives as input the `Decoder-to-Grapheme`'s probabilistic forecasts of the next character, and it keeps track of the history of predictions to feed back to `Decoder-to-Grapheme`. Because `Decoder-to-Grapheme` operates as an auto-regressive model at inference, by tweaking the history the `Post-Processor` feeds back, we can generate more than a candidate quasi-homophone. We use a Beam Search algorithm to find the best candidates.

At each inference step, the `Post-Processor` stores the C most-likely predictions of the `Decoder-to-Grapheme`, and it constructs alternative histories to feed back to `Decoder-to-Grapheme` at the following inference step. Figure 3 depicts this iterative process in the form of a tree (with $C = 2$), where each edge has an associate probability. It starts from the IPA representation of the word `by`, and each node represents the two most likely next-character given

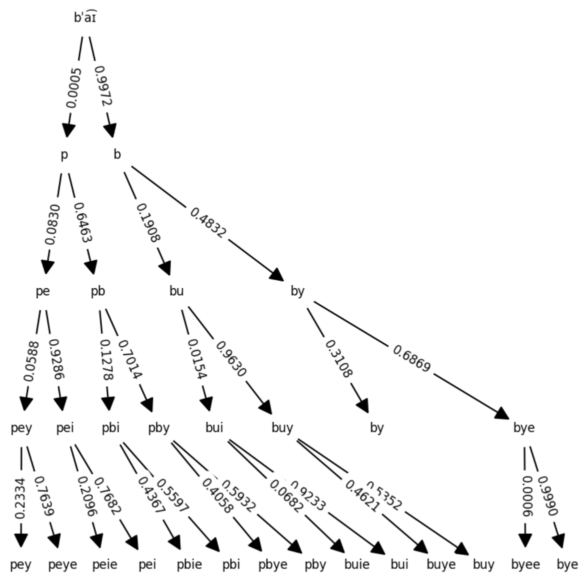


Figure 3: Illustration of the inference process with $K = 2$ children per node. At each inference step, we collect the two next characters with the highest probability. The left and the right children have the second-highest and highest probability, respectively. Some nodes do not have children because they reach the "End of Sentence" state.

the past sequence of characters. The letter b is the best candidate first letter (with probability 99.72%), while p results the second-highest. b can then be followed by u or y, etc. After repeating the exploration four times, the process generates 15 different ways to write by. Notice that the Post-Processor stops the generation when the Decoder-to-Grapheme outputs the special character EoS (End of Sentence). This happens for instance when generating by.

To find the best quasi-homophone candidates, Post-Processor computes the joint probability of each leaf as being the product of the edge probabilities. At each step, Post-Processor ranks current leaves by joint probabilities and keeps the top-K most likely ones to prune the search space and avoid pursuing branches with a low likelihood of producing good quasi-homophones. The number of iterations M at which stop the generation process, the number of candidate predictions (children) C to generate at each step, as well as the number of best candidates K to keep are additional hyper-parameters of Sound-skwatter which can be either defined manually or identified using standard local-search procedures. For the purpose of this work, given N the length of the input word, Post-Processor iterates $M = N + 6$ times, generating potentially C^{N+6} candidates. To limit the exploration, we set $K = 64$ and choose $C = 2$. These parameters have been determined by manual inspection, and it is clearly heavily dependent on the tasks and datasets of interest.

4 SOUND-SKWATTER: TRAINING AND VALIDATION

After defining the model for generating homophones and quasi-homophones, we describe the language to which we will apply our model. We select American English since it is the most used language for digital services and on the Internet. English is also phonetically inconsistent, thus a potential target for squatting. Recall, however, that the model and methodology are language-independent, and Sound-skwatter can be trained for any language. We collect a training set of words, IPA pronunciation, and Mel Spectrogram to train Sound-skwatter (Section 4.1). We then validate the trained Sound-skwatter using a list of known homophones and check if it can generate them (Section 4.2).

4.1 Dataset and training

The dataset used during training contains American English words. We consider the list of words from the GNU Aspell [3] reference word list. GNU Aspell is a free and open-source spell checker and contains 125 929 words in total for American English.

To acquire the pronunciation, we use the open-source software eSpeak NG Text-to-Speech. It supports more than 100 languages, and we set it for American English to solve the grapheme-to-phoneme translation and to produce the speech sound signal. The generated speech sounds artificial to a human ear, but it is adequate for our goals.

The final step is to convert the sound signal to a spectrogram at the Mel Scale. We use Torch Audio library [29] to produce those.

We train the model with a batch size of 64 words. The training set contains 80% of the samples, and we preserve 10% for both the validation and the test sets. We use the validation set to choose the best model and the test set to verify overfitting. We use the Adam optimizer with $LR = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. We set the step learning rate decay with $\gamma = 0.1$ scheduled every 10 epochs. We train the model for 30 epochs (about 47k steps) which took around 168 minutes on a single Nvidia Tesla v100. We train Duration-Predictor with the same dataset and the same Adam optimizer. Duration-Predictor converges at around 100k steps. Further details regarding the training are found in Appendix A.

4.2 Validation

We verify if the model can generate homophones by selecting known sets of homophones for American English. We consider as homophones any group of words having the same IPA encoding. Then, we choose one word for each set and ask Sound-skwatter to generate candidate squatting words. We expect a considerable intersection between the set of known homophones and the set of generate candidates.

For a set of 125 929 words in American English, we found 2 279 sets of homophones for a total amount of 5 804 homophones. In the validation process, we generate 47 024 candidates: on average, we produce 18.49 candidates per target, with a standard deviation of 7.61. The intersection size between the candidates and the known homophones is 4 870, i.e., Sound-skwatter found 83.91% of the possible homophones. We consider this a very good coverage, allowing us to conclude that Sound-skwatter can generate words with exact and similar pronunciation from a target. We show some examples

Table 2: Evaluation of the capability to generate homophones using as control the known homophones in American English. The last column contains the candidates ordered by the probability of being correct from the highest to the smallest. Notice that the known homophones are at the beginning of the list because they are, in fact, the best candidates for the given pronunciation.

Word	Known homophones	Candidates Ordered by Joint Probability
hobbes	hobbs, hobs	**hobbs**, **hobs**, haabs, hobbes, haabs, hobbes, ho...
niche	nitsche, nitsch	nitch, **nitsch**, nich, **nitsche**, knich, niche, k...
gainer	gaynor, gayner, gainor	gainer, **gainor**, **gayner**, **gaynor**, gainer, gayne...

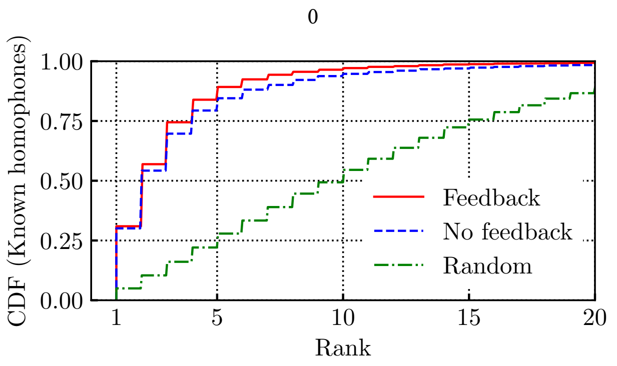


Figure 4: The ECDF for the rank in which we find the known homophones on the list of generated candidates when sorted by joint probability. As the ECDF is substantially higher than random ordering, we can conclude that using the joint probabilities is a valid method.

in Table 2. In addition, Sound-skwatter generates possible alternate spelling of the same target work, introducing several degrees of difference. Candidates are ordered according to the joint probability. Notice that homophones (in bold) appear among the top candidates.

Checking if the generated words are existing in the American English vocabulary, we find that out of 47 024 candidates, 1 975 are actual words, i.e., they are quasi-homophones.[4] The remaining words are not present in the dictionary but show certain similarities with the target word.

To confirm that the joint probability is a valid method to rank the candidates, Figure 4 shows the Empirical Cumulative Distribution Function (ECDF) curve that represents the probability of finding a homophone in the top n-th elements in the list. The red curve refers to sorting by the joint probability. The green curve refers to random sorting. Notice that more than 80% of the homophones are found in the top 5 candidates, a substantially higher proportion than a random choice.

4.3 Impact of acoustic feedback

Here we compare Sound-skwatter with a simpler model where we omit the `Duration-Predictor` and the `Decoder-to-Mel`. While simpler, it is also limited in the cross-language case since the audio part plays an important role in such case.

[4]We use Enchant Spellchecker dictionary to verify if the word exists in the US English vocabulary.

Comparing the results on the known English homophone test, we observe that Sound-skwatter generates 4 870 with sound feedback and 4748 without. The different is 122. However, we see a bigger difference when we compare the unseen homophones and quasi-homophones, with additional 938 words found including the acoustic feedback.

For completeness, we report in Figure 4 the ECDF curve that represents the probability of finding a homophone in the top n-th elements in the list for the model without acoustic feedback. The curve for the model without feedback in blue is to the right of the red curve indicating a lower quality in the generation because the known homophones are found later in the rank of generated candidates.

Some examples are shown in Table 3. We notice that the generated samples using the feedback capture more subtle sound similarities, which is good for sound-squatting applications. For instance, without sound feedback, the model it is not capable to assign to the word 'silvester' homophones ending in with the syllable 'tre' as 'silves**tre**', and 'sylves**tre**'. This is a subtle similarity that the acoustic feedback enables during the generation. The addition of acoustic features bring a qualitative improvement, without performance penalties.

5 CASE STUDIES

In this section, we explore possible scenarios where sound-squatting may be a threat and show how Sound-skwatter can be used to prevent abuses. Our goal is to demonstrate that Sound-skwatter can be used for proactive protection in any context that requires names as identifiers. We do not intend to present an extensive study on the possible abuse of sound-squatting in the wild, and we limit to illustrate that this type of attack is possible and Sound-skwatter can help to mitigate it.

Armed with a list of candidates, one can apply standard solutions to protect the original target. These solutions include blocklisting, preventive registrations, showing warnings to the users, etc.

5.1 Same- and Cross-language sound-squatting

When asking Sound-skwatter to generate a homophone, we have considered the case of same-language sound-squatting, i.e., the legitimate and the sound-squatting candidates derive from the same language. This is the case of, e.g., an English speaker that has to write a word while listening to an English reader uttering the word.

Cross-language sound-squatting explores the pronunciation similarity between words in different languages instead. Confusion might occur in any combination of languages and can be a means of creating effective sound-squatting phishing campaigns. The attack

Table 3: Evaluation of the increment of candidates added by the increment of a sound feedback to the model. Preliminary results show that we have an increase of variability in the generation.

Word	Known homophones	Gen. With Sound Feedback	Gen. No Sound Feedback
pauley	pauli, pawley, paulie	paulie, pauli, pawley	pawley
chae	quaye, kea, ke, quay, key, keye, khe, kee	kee, key, quaye, keye, quay	ke, kea, quay, quaye
ais	uys, eis, eyes, ise, ayes	ise, eyes, eis	ise
heines	heinz, heinze, heins, hines, hynes	heinze, heinz, hines	heins, hines
silvester	sylvester, silvestre, sylvestre	silvestre, sylvester, sylvestre	sylvester

Table 4: Examples of same- and cross-language sound-squatting for different languages. The possible pronunciation approximations that Sound-skwatter generates are ordered according to their joint probability.

Language *A*	Language *B*	Grapheme	Cross-language homophones
US English	US English	community	community, komunity, kommunity, comunity, kemunity, chommunity, ...
US English	Italian	comunità	communita, komunita, communeta, comunita, kemunita, kommunita, ...
US English	Portuguese	comunidade	communidade, communidayed, communidayde, communedayed, ...
US English	Spanish	comunidad	communidad, communedad, communidade, komunidad, komuniadad, ...

can be highly effective if attackers focus on regions (and groups of people) where brand owners neglect protection measures.

Cross-language squatting might happen in different situations, such as:

- **Someone reads a foreign word to a second person who writes it in their own language**. A speaker of a language *A* reads some word from a language *B*, pronouncing the grapheme according to language *A* rules and understanding/transcribing the word wrongly.
- **Someone listens to a foreign word and writes it in her native language spelling**. A speaker of a language *A* listens to the pronunciation of some word from a language B and phonetically tries to write the word in her own language;

Consider, for example, the Portuguese word *"milho."* In the first case, *"milho"*, when read by an American English speaker, would be similar to *"millhoe."* This attack can be used together with smart-speakers and Text-to-Speech tools too, where the word is miss-pronounced and induce the device to make a mistake, e.g., trick a smart-speaker in: "Alexa, access to `millhoe.com`" but, actually the user wants to go to `milho.com`. To mimic this type of attack, Sound-skwatter needs to get as input the foreign language word grapheme form. The `Grapheme-to-Phoneme` would "read" it as if it was English, and the rest of the architecture would generate possible grapheme forms an English writer could produce.

In the second case, the *"lh"* in Portuguese sounds like the English *"ly."* An English listener listening to the Portuguese word would understand the pronunciation as something like "mee-lee-yo." An attacker could thus create this cross-language homophone *"meeleeyo"* and use it to trick users into accessing to `meeleeyo.com` instead of `milho.com`.

In Sound-skwatter, the `Grapheme-to-Phoneme` is configured to read the input word in Portuguese, giving its IPA representation as output. The `IPA-Encoder`, trained with English, will then interpret the IPA phonemes as an English listener, and the `Decoder-to-Grapheme` would generate possible English grapheme forms for the target word. Notice here that the `IPA-Encoder` must be able to handle the case of phonemes that are not present in the destination language but in the original language. A simple replacement with the closest phoneme policy would suffice.

We show some examples of how Sound-skwatter can automatize the creation of cross-language squatting attacks of the first case. We train Sound-skwatter to emulate how an American English reader would read and then write the grapheme form of words she reads from a foreign language.

As language B, we select three Latin languages: Italian, Portuguese and Spanish. Consider the word *"community."* It has similar grapheme forms for all languages — see Table 4. In this experiment, we ask Sound-skwatter to generate homophone candidates in language *A* mimicking how a language *A* speaker would pronounce the original word in language *B*.

Table 4 lists examples of words that might induce a user to errors. The possible pronunciation approximations that Sound-skwatter generates are ordered according to their joint probability. The first line shows the same-language squatting for completeness, i.e., in American English.

Here Sound-skwatter outputs the correct form to write "community" as the first word, which is the one with the highest probability. Then it proposes a list of possible candidate homophones. For Italian, the correct way of writing "comunità" is not even present in the list, i.e., Sound-skwatter mimics an American English reader that would likely not be able to write such a word correctly. The same happens in other languages.

In the following, we use Sound-skwatter to automatically generate candidates starting from words taken from real use cases. We consider the domain name squatting and Python packages sound-squatting, i.e., someone registering domains that may sound similar to a given target domain or submitting Python packages with similar names to popular packages.

5.2 Domain Name Sound-Squatting

Domain squatting is a technique that attackers use when they register perceptively confusing domain names aiming at tricking visitors into them [30]. Among the different squatting types, domain squatting based on sound-squatting has received little attention while gaining traction with the advent of smart speakers and voice-assistants [12].

Here we present a thorough analysis for domain sound-squatting using Sound-skwatter to produce quasi-homophones. Given target domains, we generate sets of squatting candidates and check if such domains exist. The process is as follows: (i) we collect a list of domains from target services, (ii) we extract the Second Level Domain names, as the portion of the domains susceptible to abuse; (iii) we run the domains over Sound-skwatter and collect the generated candidates. Sound-skwatter is trained as an American English speaker listening to the domains in English and writing them in English. If the original target domains are English words, Sound-skwatter generates same-language candidates. If the original domain is of a different language, Sound-skwatter generates cross-language candidates.

Given the list of candidates, we verify the quality of our generation by assessing if any of these domains exist. The process is as follows: (i) We form fully qualified domain names by concatenating the quasi-homophones to a Top Level Domains (TLDs); (ii) We verify if such domains are actually registered; (iii) If a domain exists, we query external sources to classify the domain as **Malicious** or **Suspicious**. For this last step, we use Google Safe Browsing to mark the domain as Malicious and Virus Total to flag it as Malicious or Suspicious.[5] If the domain results are not present in any lists, we check if it could be a **Target-Owned** domain, i.e., a domain registered by the same company that owns the target domain. This protection mechanism is usual for companies and brands concerned about their reputation. Usually, these Target-Owned domains redirect the user to the correct domain when accessed. To detect if the candidate domains belong to the Target-Owned class, we compare the information registered in the `Whois` database for the target and candidate domains. If the owner is the same, and it is a first-party identifiable from the brand (i.e., not a third-party service), we declare the candidate domain as Target-Owned. Finally, we classify as **Unknown** the remaining candidates.

5.2.1 Experiment design and results. For this experiment, we select the 200 most popular websites from the Similar Web list [14]. This list includes domains in any language, thus leading to both same- and cross-language cases. After extracting only the Second Level Domains, we have 164 unique words that we consider as our targets.[6] We generate candidates for each of them, asking Sound-skwatter to output at most the $K = 64$ most likely domains according to their joint probability. As shown in Table 5, in total we obtain 3 681 candidates,[7] an average of 22.44 by target, with standard deviation of 11.60.

We next consider the 10 most popular TLDs again according to Similar Web. They are: *.com*, *.ca*, *.org*, *.ru*, *.net*, *.au*, *.uk*, *.in*, *.ir*, and *.de*. The product between any target and any TLD produces 36 810 candidate domains, out of which 3 692 results are registered. This number represents 10.03% of the total.

We classify the existing candidates by first searching at Virus Total and Google Safe Browsing blocklist. The Table 6 summarizes the results. The search on Google Safe Browsing offers surprisingly lacking results with only 4 candidates being present in the blocking list, i.e., telegram: `telegragm.com`, shopee: `shhopee.com`, as: `az.org`, cnbc: `snbc.com`. All the other 3 677 candidate domains are not present. The search on Virus Total shows more results. We find 237 domains to be reported as Malicious and 69 as Suspicious by, at least, one of the Security Vendors consulted by the platform. 485 are not existing in the Virus Total database, while the remaining 2 722 are present but not flagged by the platform.

Using `Whois`, we check the owners of those candidates that are not flagged. We find several examples owned by companies known to be proxy domain register services like: Anonymize, Inc; Domains By Proxy, LLC; and PrivacyProtect.org. These services are a common practice for squatting domains via parking or hiding the actual identity of the domain owner. Manual checking some of those, we indeed observe a large majority of parking domains.

The information from `Whois` let us automatically classify 37 cases as Target-Owned domains. We observe companies such as Microsoft Corporation, Meta Platforms, Netflix, and TripAdvisor that proactively register some possibly confusing names. For instance, we found 16 domains among those we generate to be owned by Microsoft, including `mikrosoft.com` and `sharepointe.com`. Table 7 show some examples for other brands. Appendix B completes the list and provides additional details. Notice how Sound-skwatter is able to include extra symbols, e.g., the "-" in `share-point.com` that represents a silence in the compound work. We also find 16 possible Target-Owned, but we do not have enough data to support the assumption. We list these domains in Table 12.

Table 5: Summary of sound-squatting for Domain names.

	Quantity
Initial Domains	200
Unique Domains	164
Generated Candidates	3 681
Candidate Domains	36 810
Existing Candidates	3 692 (10.03%)

Table 6: Classification of candidate Domain names that exist.

	Virus Total	**Safe Browsing**	**Whois**
Not Found	485	–	–
Target-Owned	–	–	37
Malicious	237	4	–
Suspicious	69	–	–
Unknown	2 738	3 693	–

[5]Virus Total uses a reputation mechanism to flag domains as Suspicious or Malicious.
[6]We handle the case of meaningless second-level domains, e.g., `.co.uk`
[7]Naturally, this number excludes the original target name, if Sound-skwatter outputs the target as illustrated in Table 2.

Table 7: Brands that proactively register confusing domains to increase the reputation of the brand and avoid squatting.

Target	Example	Owner	# Owned Domain
microsoft	mikrosoft.com	Microsoft Corporation	7
sharepoint	share-point.com	Microsoft Corporation	9
netflix	netflics.com	Netflix, Inc.	4
tripadvisor	trypadvisor.com	TripAdvisor LLC	4
facebook	facebuk.com	Meta Platforms, Inc.	1

Table 8: Summary Python Package analysis.

	Quantity
Original Packages	4 299
Generated Candidates	177 998
	Mean = 41.40
	SD = 19.41
Existing Candidates	1 176

Table 9: Examples of PyPI candidates present in the repository.

Target	Candidate	Class
flask	flasque	Empty project
pandas	panndas	Joke
quandl	kwandl	Legitimate
sphinx	sfinx	Possible joke
unify	uniphi	Empty project

5.3 Sound-Squatting at Python Packages

The Python Package Index (PyPI) is a software repository for the Python programming language. PyPI allows users to search packages by name, keywords, and metadata. Several package managers use PyPI as the primary source for packages and dependencies. The ease offered by these package managers makes them susceptible to squatting, e.g., an attacker could register malicious packages with confusing names to impersonate popular packages. This attack has already happened in the past. For instance, `jeIlyfish` and `python3-dateutil` impersonated the `jellyfish` and the `dateutil` packages, respectively. They use typo- and combo-squatting, respectively. If installed, the malicious versions would steal SSH and GPG keys and send them to a remote server. Both packages have been removed from the PyPI repository.

An attacker could also use sound-squatting to trick users into downloading malicious packages. We investigate the possibility of the attack by collecting popular package names, generating candidates for each of them, and verifying the existence of these candidates in the official PyPI repository. The mere existence of a candidate does not prove that the package is malicious — whose actual identification would be cumbersome and out of the scope of this work. However, it raises some concerns about the exposure of the original package to sound-squatting. Here we focus on some interesting cases.

We consider the packages with the highest number of downloads from the official PyPI repository.[8] Qualitatively, these package names present complicated patterns with dashes, version numbers, and abbreviations that challenge the generation of candidates. Yet, Sound-skwatter can generate qualitatively good-quality candidates without the need to specialize it to the specific use case.

For this analysis, we select the 5 000 most downloaded packages. After removing the packages with more than 20 characters,[9] we count 4 299 unique packages. We run the candidate generation for each package and obtain 177 998 candidate names, which we verify if they exist at the PyPI repository.

[8]The number of downloads is not an indisputable metric. However, it is enough for this analysis.

[9]This is the largest length our model currently accepts.

As shown in Table 8, we found 1 176 candidates that exist (about 0.66% of the total of candidates). Of the 4 299 original packages, 719 of them (16.72%) have at least one alternative in the Sound-skwatter candidates, which might generate confusion amount users. Table 9 gives some examples. By manual inspection, we find some empty projects that may become malicious in the future (as for parking domains), some joke packages, and many legitimate packages.

In a nutshell, the presence of a package does not indicate maliciousness. However, it exposes a threat that needs to be further considered. For instance, when checking some statistics that correlate to the quality of the candidate packages, we find that 522 existing candidates have 0 stars and 0 forks. These packages are good starting points for verification, and the maintainers of the target projects could evaluate if they are victims of sound-squatting. Similarly, tools using PyPI could warn a user when she tries to install one of the possible sound-squatted packages. Sound-skwatter offers automatic means to generate such cases.

6 RELATED WORK

Nikiforakis et al. present the first work that uncovers sound-squatting as a domain name phishing technique [16]. They propose a tool for generating sound-squatting candidates that uses a static database of homophones for English. They split the domain names into words and replace the words with homophones. After generating candidates for a list of popular domains, the authors exhaustively evaluate and categorize the candidates. The authors have also registered sound-squatting candidates, evaluating the attracted traffic to prove that sound-squatting is a possible attack vector. With Sound-skwatter, we build on AI techniques to automatically create sound-squatting, obtaining a much more extensive list of candidates. We compare a simplified version of this work [10] with our model at the Section4.2.

[10]Since no source code was provided

In [18] the authors introduce an initial AI-based sound-squatting generation approach. They train a Transformers Neural Network to translate from English to IPA, introduce some noise in the process, and translate back from IPA to English to obtain sound-squatting candidates. Sound-skwatter improves this initial idea. First, it avoids the explicit introduction of noise, which could bias the quality of the generated candidates. Second, Sound-skwatter avoids the translation to/from IPA and builds on the actual pronunciations with the Transformers architecture. Such changes make Sound-skwatter a general approach applicable to any language and cross-language sound-squatting scenarios.

Considering phoneme-based squatting, Sonowal and Kuppusamy focus on visually impaired people, typo-squatting, and phoneme-based phishing [23]. Their motivation is that visually impaired people use software to read the screen and interact with the system. They propose a system to detect phoneme-based phishing domains within the accessibility interface. In a second work, Sonowal evaluates the exposure of visually impaired people to phoneme-based phishing in email phishing campaigns [22]. He proposes a manual and simplistic technique to produce sound-alike keywords, using these keywords to detect phishing emails.

Considering speech, Kumar et al. uncover the *skill squatting* attack, where the attacker leverages systematic errors to route users of Alexa Smart Speakers to malicious applications [12]. They show that around 33.3% of the systematic errors in the speech-to-text system are due to homophones. Later, Zhang et al. formalize the skill squatting attack, calling it Voice Squatting Attack (VSA) and Voice Masquerading Attack (VMA) [31]. They also evaluate the feasibility of such attacks by deploying a malicious skill, which has been invoked by 2 699 users in a month.

Sound-skwatter is orthogonal to all these efforts. It offers the automatic means to generate sound-squatting candidates. By combining the Transformers models with sound models, Sound-skwatter works both in the same- and cross-language cases, widening the possibility of a proactive defense against sound-squatting attacks.

AI approaches, such as Transformers Neural Networks, have been used to generate artificial content such as text [19], images [9, 17] and music [8]. These previous efforts however focus on legitimate content, where the goal is to produce realistic and correct samples. Sound-skwatter instead aims at generating content that can confuse people. Authors of [15]—focusing on image generation—propose a way to create samples that can be understood but are perceptively confusing to people. Analogously, Sound-skwatter generates words that are perceptively confusing regarding their pronunciation. In other words, we want our generation to induce perception errors and rely on a novel approach that introduces errors at a sub-word level.

7 CONCLUSION

In this work, we introduced Sound-skwatter, an AI-powered sound-squatting generator. It uses Transformers Neural Networks trained to model both the phoneme representation and the pronunciation of words, producing high-quality homophones, quasi-homophones, and novel words with similar pronunciation. Because Sound-skwatter generates sound-squatting candidates at the sub-word level, it increases the coverage while still allowing one to control the number and quality of the candidates.

We also laid the ground for cross-language sound-squatting search and mitigation. We showed that Sound-skwatter could natively work not only with multiple languages but also cross-languages. Finally, we thoroughly analyzed Sound-skwatter in two contexts, showing that sound-squatting is likely exploited for domain squatting already. We showed that multiple popular PyPI packages are exposed to squatting attack.

Considering in more detail the domain names case study, we found a significant number of malicious and suspicious domains already reported at Virus Total while revealing a surprising lack of coverage for Google Safe Browsing. We also found a reasonable amount of Target-Owned domains preemptively registered by companies concerned about their reputation. In the remaining analyzed domains, we found high occurrences of domains registered in proxies, indicating some attempts to obfuscate the property.

For future work, we believe there is potential to increase the coverage for other applications, such as sound-squatting for smart-speakers routines, applications on app stores, and profiles on social networks. We also think the Sound-skwatter has the potential to increase the depth of analysis and go beyond the surface of the use cases already explored.

Another possible angle of improvement is the cross-language sound-squatting, which needs a uniform approach and a thorough evaluation of its usage, including the case where smart speakers and other Speech-to-Text solutions leverage it.

REFERENCES

[1] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick Nikiforakis. 2015. Seven months' worth of mistakes: A longitudinal study of typosquatting abuse. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society.

[2] The International Phonetic Association. 2022. *The International Phonetic Association Homepage*. https://www.internationalphoneticassociation.org/

[3] Kevin Atkinson. 2006. Gnu aspell 0.60. 4.

[4] MITRE ATT&CK. 2022. CAPEC-616: Establish Rogue Location. https://capec.mitre.org/data/definitions/616.html.

[5] MITRE ATT&CK. 2022. CAPEC-631: SoundSquatting. https://capec.mitre.org/data/definitions/631.html.

[6] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. Association for Computing Machinery, New York, NY, USA, 369–376. https://doi.org/10.1145/1143844.1143891

[7] Tobias Holgers, David E Watson, and Steven D Gribble. 2006. Cutting through the Confusion: A Measurement Study of Homograph Attacks.. In *USENIX Annual Technical Conference, General Track*. 261–266.

[8] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music Transformer. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJe4ShAcF7

[9] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. 2021. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074* 1, 3 (2021).

[10] Mohammad Taha Khan, Xiang Huo, Zhou Li, and Chris Kanich. 2015. Every second counts: Quantifying the negative externalities of cybercrime via typosquatting. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 135–150.

[11] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero-Gómez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. 2017. Hiding in plain sight: A longitudinal study of combosquatting abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 569–586.

[12] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill squatting attacks on Amazon Alexa. In *27th USENIX security symposium (USENIX Security 18)*. 33–47.

[13] Pablo Loyola, Kugamoorthy Gajananan, Hirokuni Kitahara, Yuji Watanabe, and Fumiko Satoh. 2020. Automating Domain Squatting Detection Using Representation Learning. In *2020 IEEE International Conference on Big Data (Big Data)*. 1021–1030. https://doi.org/10.1109/BigData50022.2020.9377875
[14] Similarweb LTD. 2022. Similarweb - Check and Analyse any website. https://www.similarweb.com/.
[15] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. (2015).
[16] Nick Nikiforakis, Marco Balduzzi, Lieven Desmet, Frank Piessens, and Wouter Joosen. 2014. Soundsquatting: Uncovering the use of homophones in domain squatting. In *International Conference on Information Security*. Springer, 291–308.
[17] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image Transformer. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 4055–4064. https://proceedings.mlr.press/v80/parmar18a.html
[18] Authors Removed. 2022. Removed to adhere to the double blind policy. In *Booktitle*. 1–6.
[19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. https://doi.org/10.48550/ARXIV.1910.01108
[20] John Semmlow. 2018. Chapter 6 - Linear Systems in the Frequency Domain: The Transfer Function. In *Circuits, Signals and Systems for Bioengineers (Third Edition)* (third edition ed.), John Semmlow (Ed.). Academic Press, 245–294. https://doi.org/10.1016/B978-0-12-809395-5.00006-0
[21] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 4779–4783.
[22] Gunikhan Sonowal. 2020. A Model for Detecting Sounds-alike Phishing Email Contents for Persons with Visual Impairments. In *2020 Sixth International Conference on e-Learning (econf)*. IEEE, 17–21.
[23] G Sonowal and KS Kuppusamy. 2019. MMSPhiD: A Phoneme based Phishing Verification Model for Persons with Visual Impairments. Information and Computer Security Journal.
[24] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. 2014. The Long {"Taile"} of Typosquatting Domain Names. In *23rd USENIX Security Symposium (USENIX Security 14)*. 191–206.
[25] Ke Tian, Steve T. K. Jan, Hang Hu, Danfeng Yao, and Gang Wang. 2018. Needle in a Haystack: Tracking Down Elite Phishing Domains in the Wild. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. Association for Computing Machinery, New York, NY, USA, 429–442. https://doi.org/10.1145/3278532.3278569
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[27] Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski, and Brad Daniels. 2006. Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. *SRUTI* 6, 31-36 (2006), 2–2.
[28] Dyslexia Reading Well. 2022. *The 44 Phonemes in English.* https://www.dyslexia-reading-well.com/44-phonemes-in-english.html
[29] Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. 2021. TorchAudio: Building Blocks for Audio and Speech Processing. *arXiv preprint arXiv:2110.15018* (2021).
[30] Yuwei Zeng, Tianning Zang, Yongzheng Zhang, Xunxun Chen, and YiPeng Wang. 2019. A Comprehensive Measurement Study of Domain-Squatting Abuse. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC.2019.8761388
[31] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2019. Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1381–1396.

A ARCHITECTURAL DETAILS

Figure 5 details the architecture of the model used for homophone generation. We include in the diagram the `IPA-Encoder`, `Decoder-to-Grapheme` and `Decoder-to-Mel`. We also show in Figure 6a the inside of the decoder block that we used to reduce the complexity of the visualization. Figure 6b is a high-level representation of the Length Regulator. The Length Regulator expands the `IPA-Encoder` output to the same order of magnitude as the Mel Spectrogram. This feature reduces training complexity and time. Table 10 lists the hyper-parameters.

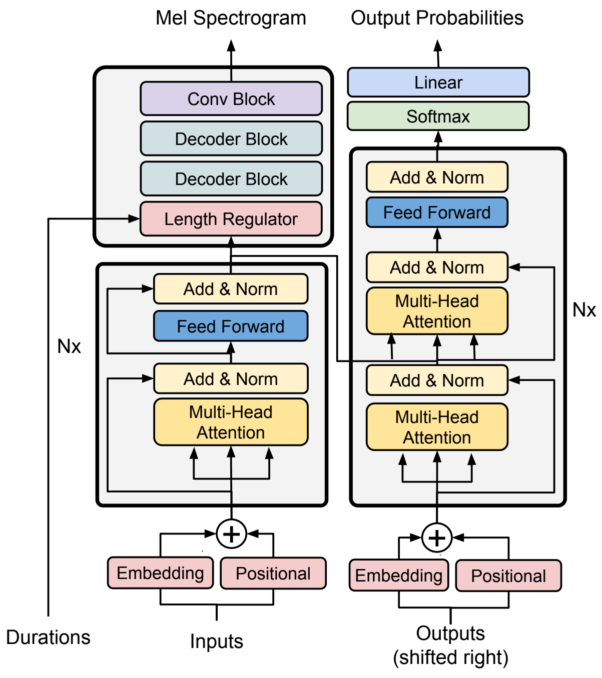


Figure 5: Full architecture of training

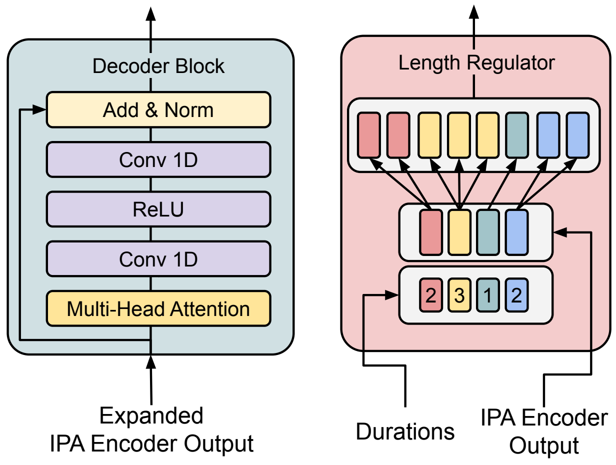


Figure 6: (a) Inside view of the Decoder Block; (b) High-level representation of the Length Regulator.

Target	Candidate	Owner
facebook	facebuk.com	Meta Platforms, Inc.
yandex	yandx.ru	YANDEX
wikipedia	wiki-pedia.org	Wikimedia Foundation.
netflix	netflicks.com	Netflix, Inc.
netflix	netflicks.org	Netflix, Inc.
netflix	netflicks.net	Netflix, Inc.
netflix	netflics.com	Netflix, Inc.
naver	navor.com	NAVER Corp.
microsoft	microsopft.com	Microsoft Corporation
microsoft	mikrosoft.com	Microsoft Corporation
microsoft	microsofte.com	Microsoft Corporation
microsoft	microsofte.net	Microsoft Corporation
microsoft	michrosoft.com	Microsoft Corporation
microsoft	mikerosoft.com	Microsoft Corporation
microsoft	mikerosoft.net	Microsoft Corporation
quora	kwora.com	Quora, Inc
quora	quorea.com	Quora, Inc
sharepoint	share-point.com	Microsoft Corporation
sharepoint	share-point.org	Microsoft Corporation
sharepoint	share-point.net	Microsoft Corporation
sharepoint	sharepointe.com	Microsoft Corporation
sharepoint	sharepointe.org	Microsoft Corporation
sharepoint	sharepointe.net	Microsoft Corporation
sharepoint	share-pointe.com	Microsoft Corporation
sharepoint	share-pointe.org	Microsoft Corporation
sharepoint	share-pointe.net	Microsoft Corporation
target	targett.org	Target Brands, Inc.
target	taarget.org	Target Brands, Inc.
tripadvisor	tripadviser.com	TripAdvisor LLC
tripadvisor	tripadviser.org	TripAdvisor LLC
tripadvisor	tripadviser.net	TripAdvisor LLC
tripadvisor	trypadvisor.com	TripAdvisor LLC
hulu	hoolu.com	Hulu, LLC
hulu	hoolu.org	Hulu, LLC
hulu	hoolu.net	Hulu, LLC
hulu	huloo.org	Hulu, LLC
hulu	huloo.net	Hulu, LLC

Table 11: Full compilation of found Target-Owned domains.

Target	Candidate	Owner
spankbang	spankbank.org	Redacted for Privacy
stripchat	stripchatt.com	Redacted for Privacy
pornhub	pornhubb.com	Whois Privacy
pornhub	porn-hub.org	Whois Privacy
pornhub	pornhab.com	Whois Privacy
wildberries	wildberris.ru	Private Person
wildberries	wildberies.ru	Private Person
zhihu	zhihoo.com	Redacted for Privacy
deepl	deepaul.com	Redacted for Privacy
onlyfans	onlyphans.net	Domains By Proxy, LLC
telegram	telegramm.org	Domains By Proxy, LLC
telegram	telegramme.org	Domains By Proxy, LLC
telegram	tellegram.com	Domains By Proxy, LLC
telegram	telligram.com	Domains By Proxy, LLC
flipkart	flipcart.net	Domains By Proxy, LLC
flipkart	flypkart.com	Domains By Proxy, LLC

Table 12: Other Target-Owned compilation, however owned by proxy and private individuals.

Table 10: Compilation of hyper-parameters for Sound-skwatter

Hyper-parameter	Value
# Phoneme Tokens	73
# Grapheme Tokens	33
Phoneme Embedding Dimension	512
`IPA-Encoder` # of Attention Heads	2
`IPA-Encoder` Hidden Dimension	512
`IPA-Encoder` Linear Hidden	2048
`IPA-Encoder` Dropout	0.1
`Decoder-to-Grapheme` # of Attention Heads	2
`Decoder-to-Grapheme` Hidden Dimension	512
`Decoder-to-Grapheme` Linear Hidden	2048
`Decoder-to-Grapheme` Dropout	0.1
`Decoder-to-Mel` Decoder Block Number Decoder Block	2
`Decoder-to-Mel` Decoder Block Attention Hidden Dim	512
`Decoder-to-Mel` Decoder Block Conv1D # Kernel	9, 9
`Decoder-to-Mel` Decoder Block Conv1D # Filters	1024, 512
`Decoder-to-Mel` Conv Block Number of Conv1D	6
`Decoder-to-Mel` Conv Block Conv1D # Kernel	9
`Decoder-to-Mel` Conv Block Conv1D # Filters	512
`Decoder-to-Mel` Conv Block Output Conv1D # Kernel	9
`Decoder-to-Mel` Conv Block Output Conv1D # Filters	40
Total Number of Parameters	60.2 M

B TARGET-OWNED SOUND-SQUATTING CANDIDATES

Table 11 lists the candidate domains registered by the same company owning the original domain. This is a clear sign that Sound-skwatter can generate possible sound-squatting candidates that companies are already proactively considering for their brand protection. The Table 12 shows some examples of domains that are registered by proxy companies. Some are used for Target-Owned protection. Some are parking domains. Some are registered but not hosting any web page.