Dagli archivi il Progetto per un Albergo-Rifugio tipo in Valmalenco di Gianni Albricci e Marco Zanuso (1938)

X-squatter: AI Multilingual Generation of Cross-Language Sound-squatting

RODOLFO VIEIRA VALENTIM, Politecnico di Torino, Torino, Italy
IDILIO DRAGO, University of Turin, Torino, Italy
MARCO MELLIA, Politecnico di Torino, Torino, Italy
FEDERICO CERUTTI, University of Brescia, Brescia, Italy

Sound-squatting is a squatting technique that exploits similarities in word pronunciation to trick users into accessing malicious resources. It is an understudied threat that has gained traction with the popularity of smart speakers and audio-only content, such as podcasts. The picture gets even more complex when multiple languages are involved. We here introduce X-squatter, a multi- and cross-language AI-based system that relies on a Transformer Neural Network for generating high-quality sound-squatting candidates. We illustrate the use of X-squatter by searching for domain name squatting abuse across hundreds of millions of issued TLS certificates, alongside other squatting types. Key findings unveil that approximately 15% of generated sound-squatting candidates have associated TLS certificates, well above the prevalence of other squatting types (7%). Furthermore, we employ X-squatter to assess the potential for abuse in PyPI packages, revealing the existence of hundreds of candidates within a 3-year package history. Notably, our results suggest that the current platform checks cannot handle sound-squatting attacks, calling for better countermeasures. We believe X-squatter uncovers the usage of multilingual sound-squatting phenomena on the Internet and it is a crucial asset for proactive protection against the threat.

CCS Concepts: • **Security and privacy → Network security**; • **Networks → Network management**; **Network monitoring**

Additional Key Words and Phrases: Phishing, malicious resources, proactive defense, internet protection

ACM Reference Format:
Rodolfo Vieira Valentim, Idilio Drago, Marco Mellia, and Federico Cerutti. 2024. X-squatter: AI Multilingual Generation of Cross-Language Sound-squatting. *ACM Trans. Priv. Sec.* 27, 3, Article 21 (June 2024), 27 pages. https://doi.org/10.1145/3663569

The research leading to these results has been partly funded by the project SERICS (SEcurity and RIghts In the CyberSpace - PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union, as well as the ACRE (AI-Based Causality and Reasoning for Deceptive Assets - 2022EP2L7H) and xInternet (eXplainable Internet - 20225CETN9) projects - funded by European Union - Next Generation EU within the PRIN 2022 program (D.D. 104 - 02/02/2022 Ministero dell'Università e della Ricerca). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.
Authors' Contact Information: Rodolfo Vieira Valentim, Politecnico di Torino, Torino, Italy; e-mail: rodolfo.vieira@polito.it; Idilio Drago, University of Turin, Torino, Piemonte, Italy; e-mail: idilio.drago@unito.it; Marco Mellia, Politecnico di Torino, Torino, Piemonte, Italy; e-mail: marco.mellia@polito.it; Federico Cerutti, University of Brescia, Brescia, Lombardia, Italy; e-mail: federico.cerutti@unibs.it.

ACM 2471-2566/2024/06-ART21
https://doi.org/10.1145/3663569

1 INTRODUCTION

Cyber-squatting is a practice that tricks users into accessing malicious sites or content by relying on the similarities of words. It is applied in various contexts, including fake domains [15], phishing campaigns [37, 38], the hijacking of smart speakers [20, 50] and brand abuse. Several cyber-squatting strategies have been demonstrated in practice, including simple/frequent typos [4, 17, 40, 48], visual similarity between characters [15], and combination of common words [19, 24], leading to different attacks such as typo-squatting, bit-squatting, homograph-squatting, combo-squatting [49], skill-squatting [20], and sound-squatting [30].

Sound-squatting is a relatively recent technique that exploits pronunciations similar to legitimate names or brands to deceive users. It has garnered less attention in comparison to other forms of cyber-squatting. However, it is increasing in importance, especially with the rise of smart speakers and voice assistants [20] and the resurgence of audio-exclusive content consumption, such as podcasts. A website or product verbally advertised and misunderstood by the users as well as a careless voice search[1] can, ultimately, lead users to malicious content. This may happen even when users do not rely on voice assistants—for example, consider an instructor telling students to access a website or install software packages, whose names may be mistyped due to similar pronunciations. In general, sound-squatting can be a channel for phishing attacks, with URLs or package names (and locators in general) that look and sound like the correct target but are not.

Detecting and preventing sound-squatting is challenging due to the inherent variations in pronunciation across different languages and among individuals [49]. The complexity further escalates when multiple languages are involved expanding sound-squatting possibilities, e.g., when a person must write, pronounce, or simply recognize a word pronounced in a foreign language—hereafter called *cross-language* scenario. Previous studies have primarily concentrated on English homophones [30], which are existing words with identical pronunciations. However, they fall short in terms of coverage, as they do not account for non-existing words, words with similar pronunciation (quasi-homophones), and cross-language scenarios.

We present X-squatter (pronounced `cross-squatter`), an AI-based system designed for the automatic generation of multi- and cross-language sound-squatting candidates. We have designed X-squatter to model any phonemic representation, enabling it to replace phoneme tokens with counterparts that sound similar.

X-squatter produces candidates from a target name. It operates at the sub-word level and includes a mechanism to control quality during the candidate search process. X-squatter builds over a Transformer Neural Network [47] architecture trained to produce candidates in any language. It receives the written form of the word (*grapheme*), its pronunciation represented using the **International Phonetic Alphabet (IPA)**, and the language the word belongs to. It produces homophones using features from pronunciation segments. In a nutshell, X-squatter uses a lightweight and small Transformer-based sequence-to-sequence model to find written alternatives with similar pronunciations in the desired target language.

X-squatter seamlessly accommodates multiple languages and facilitates cross-language scenarios. This requires ingenuity to handle the grapheme of non-existing phonemes in the target language. We here restrict the analysis to proto-Indo-European languages, given their prevalence in the Web (around 66.8% of the online content [33]). Yet, X-squatter is generic and its architecture can be applied to other language scenarios.

We validate X-squatter by measuring its capabilities to generate well-written forms of word pronunciations. Here we find that X-squatter can automatically generate all known homo-

[1]See an anecdotal example at https://www.bbb.org/article/news-releases/20523-scam-alert-using-voice-search-use-caution-when-asking-for-auto-dial-from-your-smart-device

phones for the languages we tested. Moreover, X-squatter produces thousands of additional quasi-homophones, including a fair share of known cross-language homophones.

We demonstrate the applicability of X-squatter in cybersecurity scenarios in practical use cases. We use X-squatter to compile a list of candidates for popular targets and check if any candidate corresponds to existing resources. We select as use cases (1) registered domain names and (2) Python software packages. In both cases, we show a large number of existing resources with practical evidence of abuses in some cases. While the mere existence of high numbers of these resources does not inherently indicate abuse, X-squatter eases the uncovering of suspicious cases.

We believe that X-squatter can bolster brand protection against impersonation attacks on their names. To amplify its impact as well as allow others to extend X-squatter to other language use cases, we release the code as open source.[2] The tool provides a cost-effective solution for scrutinizing squatting campaigns in less regulated markets by autonomously generating potential names that attackers might exploit or have already exploited. In summary, our contributions are as follows:

- **A methodology for multi- and cross-language sound-squatting generation.** We present a methodology for generating multi-language and cross-language sound-squatting candidates using features of phoneme tokens. This approach allows the model to handle pronunciations even for languages not included in the training process.
- **A comparison of sound-squatting with other squatting techniques.** We compare the search for sound-squatting with those for other squatting techniques generated with state-of-the-art algorithms. Our findings demonstrate that sound-squatting poses a different threat not fully covered by existing algorithms. X-squatter refines the hunting when combined with other types of squatting.
- **A study of sound-squatting attack surface in domain registration using issued TLS certificates.** We conduct a study of the sound-squatting attack surface in domain registration by analyzing candidates generated by X-squatter using around 900 million certificates collected via Certificate Transparency logs. These certificates represent around 127 million domains. Our findings show that approximately 16% of the generated candidates are registered, and around 95% of the analyzed domains have at least one registered candidate.
- **An extensive view on the sound-squatting attack surface in Python Packages.** We conduct a study using the **Python Package Index (PyPI)** by analyzing candidates generated by X-squatter searching on data that cover more than 900 days. Our findings show that 1,579 of the generated candidates correspond to registered packages that might be squatting 951 popular packages.

In Section 2 we present the background and related work. In Section 3 we describe X-squatter, and in Section 4 we validate it. Then we discuss our use cases in Section 5, discuss the limitations of our study in Section 6, and conclude the article in Section 7. In Appendix A, we provide details about X-squatter architecture.

2 BACKGROUND AND RELATED WORK

2.1 Tools for Squatting Generation

Cyber-squatting is a type of attack where malicious actors impersonate legitimate resources [7]. The phenomenon has been seen across various applications, encompassing domain names, software packages within repositories, and even voice applications. For example, domain squatting involves registering a name that may divert traffic from popular websites. Previous work [41] has

[2] https://github.com/rodolfovalentim/x-squatter

Table 1. Tools and Services for Squatting Candidate Generation and Evaluation

Tool/Service	Techniques	Unique	Exclusive	Coverage (%)
DomainFuzz [3]	Typo	3,403	175 (5%)	3.6%
URLCrazy [23]	Typo, Homograph, Bit, Sound	1,521	91 (6%)	1.6%
dnstwist [2]	Typo, Homograph, Bit	83,204	80,220 (96%)	87.4%
URLInsane [35]	Typo, Homograph, Bit, Sound	2,752	204 (7%)	2.9%
AIL [1, 32]	Typo, Homograph, Bit, Sound	13,274	9,777 (74%)	14.0%
X-squatter	Sound	941	856 (91%)	1.0%

shown that squatting, in general, is a serious threat, with more than 657 *k* domain names founded impersonating 702 popular brands in 2018.

Several tools and services are available to assist in the detection of squatting, and most of them have been introduced as a proactive measure to protect names. In general, these tools generate name modifications and check the availability of the candidate domains. In Table 1 we show a summary of popular tools and services. Each alternative employs a range of squatting techniques, including typo, bit-flip, homograph, combo, and sound-squatting.

For completeness, we briefly introduce the key differences between sound-squatting and the other squatting techniques. *Typo-squatting* primarily leverages typos to create confusingly similar names (e.g., gogle). *Homograph-squatting* exploits characters that visually resemble each other, sometimes using different alphabets (e.g., g00gle). Combo-squatting leverages combinations of words to fool people into believing a resource is legitimate (e.g., my-google). Bit-flip squatting relies on bit-flip errors that may change messages during transmission in insecure protocols.

We have conducted an analysis to assess the *redundancy* and *coverage* of squatting techniques among these tools and services. To achieve that, we have selected 12 high-profile target names and generated candidates using all alternatives in Table 1. We then compare the generated sets of unique names.

Notice how the number of candidates generated by each alternative varies. For example, dnstwist leads in the number of candidates with more than 83 *k* names, while URLCrazy produces 1,521. The coverage of each tool is reported in the *Coverage* column. This metric measures how many candidates each tool generates in the total set of generated candidates. As expected, it is proportional to the number of candidates. The table also reports the number of *exclusive* candidates, i.e., candidates that appear only in the set of the given tool. Notice how DomainsFuzz, URLCrazy, and URLInsane candidates are *not* exclusive. That is, these tools generate candidates that are also present in other sets (notably dnstwist), indicating redundancy in their capabilities when compared to alternatives.

To motivate the need for a new sound-squatting generation tool, we include in the table the results of applying X-squatter to the same 12 high-profile names. While X-squatter produces 941 candidates, this set is practically disjoint, even when compared with tools like AIL [32] that include techniques for sound-squatting generation. The 856 names generated by X-squatter (≈91% of the names it generated) are *exclusive* to our tool. We will show later that X-squatter candidates are of high quality too. This result suggests that sound-squatting has received less attention, calling for instruments to assist in the understanding of the phenomenon. Moreover, as we will see later, we have found some evidence that X-squatter candidates are exploited in the wild, thus calling for attention to the prevention of such attacks.

2.2 List-based Models for Sound-squatting

Specifically for sound-squatting two common approaches can be employed to generate candidates: list-based models and data-driven models. List-based models rely on predefined lists of known

homophones and common replacements that originate potential instances of sound-squatting. Such methods replace entire or pieces of the input name with their homophones, thus producing a new written form of the name with exact (or similar) pronunciation.

Nikiforakis et al. [30] generated potential instances of sound-squatting from a static database of English homophones, focusing on domain sound-squatting. Their approach involves substituting words in domain names with homophones. The authors then present a comprehensive evaluation and categorization of these generated candidates.

Considering voice-based attacks, Kumar et al. [20] uncover the *skill-squatting* attack, where the attacker leverages systematic errors on the understanding of homophones to route Alexa Smart Speaker users to malicious applications. They show that around 33% of the systematic errors in the speech-to-text system are due to homophones. Later, Zhang et al. [50] formalize the skill squatting attack, calling it **Voice Squatting Attack (VSA)** and **Voice Masquerading Attack (VMA)**. They also evaluate the feasibility of such attacks by deploying a malicious skill, which has been invoked by 2,699 users in a month by Alexa's Speech Recognition engine.

2.3 Data-driven Models for Sound-squatting

To the best of our knowledge, we have been the first to propose data-driven approaches to sound-squatting generation in our preliminary works [44–46].

Data-driven models leverage machine learning algorithms and require training data to learn patterns and produce alternative written forms for the names. We here classify data-driven models as audio-based, token-based, or hybrid models.

Audio-based models process the raw audio associated with words. They extract various audio features, such as pitch, intensity, and spectral characteristics, to capture distinctive acoustic patterns. The model then learns how to alter some of these features to generate distortions that lead to the desired alternative written forms (e.g., the homophones). These models suffer from the complexity of dealing with audio signals and controlling errors. In a way, they work similarly to a Speech-to-Text pipeline, where the generated text contains the sound-squatting candidates.

Token-based models operate on textual representations of names. They break down the input name into tokens, which can be either (1) normal grapheme tokens or (2) pronunciation tokens. Pronunciation tokens rely on a phonetic representation of the words, for instance, using the IPA.[3] These models then learn how to map the representation of the name back into a phonetically equivalent grapheme form (e.g., homophone or quasi-homophone). Since the used representation consists of categorical tokens, these models are not able to search for "close" tokens when needed.

Hybrid models use special representations of phonemes that encode the similarities of phonemes. There are several proposals for this type of representation such as the articulatory vector representation [29]. A key advantage of this approach is that it enables cross-language support. That is, it overcomes the problem created by the lack of particular phonemes in some languages. Indeed, such representations allow one to obtain a list of *similar* phonemes in the target language, which is much harder with token-based representations (e.g., based on the IPA).

We compare four alternative models. The architectures of the first three models are introduced in our preliminary work [45, 46] and summarized in the following. We then propose an improved

[3]The IPA represents phonemes with standard symbols [5]. Some languages are phonetically consistent, like Italian and German, where most graphemes correspond to a single phoneme. Other languages, like English, are not phonetically consistent, leading to pronunciation confusion, and are more susceptible to sound-squatting attacks. There exist solutions that translate any word in the corresponding IPA. For instance, the eSpeak NG (Next Generation) *Text-to-Speech* engine supports more than 100 languages.

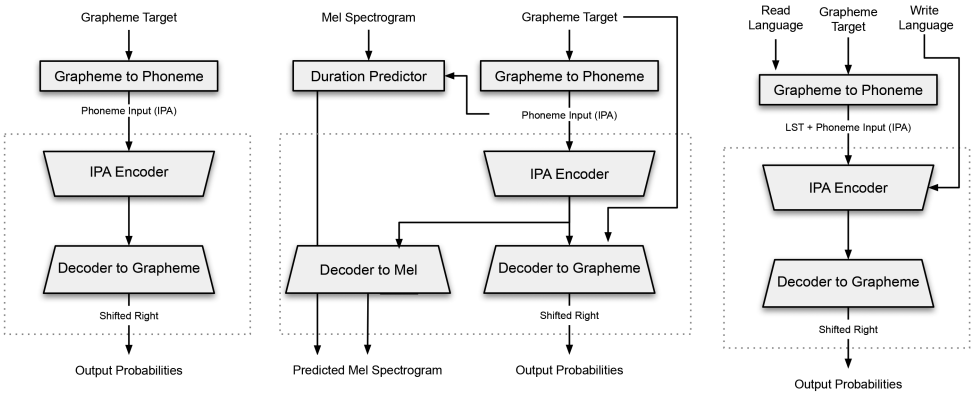


Fig. 1. Model comparison. The dotted lines mark the trainable modules. (a) Simple IPA Translation [45]. (b) Audio Inbound with IPA Translation [45]. (c) Multi-language IPA Translation [46].

architecture (X-squatter —introduced in Section 3), which generalizes the problem to handle the cross-language scenario and adds more controls for configuring the quality of candidates.

We show the training architecture of the first three alternatives in Figure 1.

(1) **Simple IPA Translation** is a token-based single-language model we introduced in [45], trained to learn transliterations from IPA segments to graphemes for a given language. This model includes *IPA Encoder* and *Decoder to Grapheme* modules and employs beam search during inference to generate multiple candidates. Since X-squatter builds upon similar ideas, we defer the details to the next section.

(2) **Audio Inbound with IPA Translation** is an audio-based single-language model (also introduced in [45]), which operates similarly to the previous model but additionally learns features from audio, thereby incorporating similarities between IPA segments.

(3) **Multi-language IPA Translation** is the first token-based multiple-language model for cyber-squatting generation, which we proposed in [46]. It features the inclusion of **special language tokens (LSTs**, which correspond to tokens representing the language ISO code [16], e.g., en-US) to generate homophones in multiple languages. However, being token based, this model cannot generate homophones in cases where there are gaps in the representation, for instance, in cross-language squatting.

In the following, we refer to these models as baselines to compare against.

2.4 List-based Versus Data-driven Models

We briefly discuss a motivation for choosing data-driven over list-based models. Consider list-based models for generating homophones. The list selection can be done by manual curation or a similarity mapping between IPA tokens. The problem is that (1) the number of existing homophones is very limited, and (2) we will miss the quasi-homophones. In fact, our evaluation of alternative approaches that rely on list-based techniques in Table 1 shows these alternatives are insufficient: they fail in terms of coverage, with X-squatter presenting 91% of uniqueness. List-based solutions miss quasi-homophones and, more importantly, they are very imprecise in the cross-language scenario, when some IPA tokens are not present in a language.

The usage of IPA as an intermediary representation helps in the generalization of list-based methods. However, IPA tokens are categorical symbols, and categorical data requires an additional

Table 2. Summary of Our Data-driven Models and Their Capabilities

Feature	Baseline 1 Simple IPA Translation	Baseline 2 Audio Inbound with IPA Translation	Baseline 3 Multi-language IPA Translation	X-squatter
Data-driven homophone generation	Yes	Yes	Yes	Yes
Multiple homophone generation	Yes	Yes	Yes	Yes
Multi-language support	No	No	Yes	Yes
Cross-language support	No	No	Partial	Yes

step to improve the representation, especially to allow us to measure similarity. In X-squatter, we overcome this problem by representing each IPA token as a set of articulatory features in a vector, which is a novel contribution over the models presented in Figure 1.

At last, sub-sequences of IPA tokens—called segments—influence each other through the speech articulation process and by the way they are grouped into syllables. Therefore, we still need to define a metric for *sequence similarity*, and finding a good similarity metric is not trivial [22]. For these reasons, the usage of Transformers is a natural choice because of their general-purpose architecture for learning a contextual function mapping sequence of symbols in sequences of symbols. It also handles peculiarities of the various languages that can be better learned from a rich dataset rather than encoded in rules.

2.5 X-squatter's Contribution over Existing Data-driven Models

We provide a qualitative example to illustrate the contribution of X-squatter over the previously mentioned methods. Let us consider the word "gnocchi," pronounced as /ˈɲɔk.ki/ in Italian. Some IPA segments in the Italian pronunciation are not present in the en-US (American English) phonetic inventory. Consequently, the Italian pronunciation of "gnocchi" encompasses sounds that do not precisely align with American English phonology.

The *Simple IPA Translation* (Figure 1(a)) and *Audio Inbound with IPA Translation* (Figure 1(b)) models cannot handle these cases, as they must be trained for a particular language. When we prompt the *Multi-language IPA Translation* (Figure 1(c)) model to generate candidates, it produces homophones based on its contextual knowledge and the IPA segments present in the target language (en-US in the example), producing "nowey," "nowy," "nori," and "norie," which are bad candidates.

X-squatter uses a hybrid approach to produce candidates for a given target. X-squatter's distinct advantage lies in its capacity to find good substitutes for IPA tokens not present in the target language. This is facilitated by the incorporation of sound features within the model, which align tokens based on articulatory attributes. Besides surpassing the limitations illustrated by the Italian-to-English example above, X-squatter's hybrid approach has the significant advantage of automatically matching token sizes (based on IPA transcripts), thus simplifying the audio signal processing when compared to pure audio-based alternatives.

In sum, X-squatter incorporates the advantages of token-based models with the flexible representation capabilities of audio-based models. Table 2 provides a comprehensive summary of our data-driven models in terms of their capabilities.

2.6 Squatting of TLS Certificates and PyPI Packages

We apply X-squatter in two use cases, covering squatting of TLS certificates and PyPI packages. The literature on attacks against both resources is vast. As we use them simply as a means to *illustrate* X-squatter in practice, we here summarize only the most significant and recent works approaching squatting in each scenario, deferring readers to these works for a complete discussion on abuses of the TLS and PyPI ecosystems.

In [18], the authors evaluate how **Certificate Authorities (CAs)** are involved in the HTTPS phishing ecosystem. The authors concentrate their study on various types of squatting, including combo-squatting, typo-squatting, and homograph-squatting. In particular, they study whether insecure practices of CAs lead to an increase in attacks. They highlight the high risk of squatted domains in the TLS environment: end-users might perceive squatted domains as legitimate when the domains are protected by TLS since they appear as *valid* in browsers' address bar. Other works [12, 36, 42] evaluate the occurrence of domain squatting using tools to create candidates and verify the existence of the created domains. However, these works focus only on techniques such as typo-squatting, combo-squatting, and homograph-squatting, ignoring sound-squatting.

Gu et al. [14] present a comprehensive analysis of security threats within software registries such as PyPI and NPM. Alongside addressing various vulnerabilities, they approach typo-squatting. The authors conclude that approximately 81.0% and 88.1% of removed packages on PyPI and NPM, respectively, share name similarities with at least one other package—i.e., they could be examples of typo-squatting. Additionally, a staggering 96.9% of the malicious packages—officially identified by NPM—have names that resemble benign packages.

3 X-SQUATTER: SYSTEM DESCRIPTION

Here we detail the neural architecture and training for X-squatter. Utilizing a Transformer Neural Network, X-squatter simplifies the process of translating phoneme feature vectors into corresponding graphemes. The functional process is shown in Figure 2(a). The generation pipeline is built upon four essential components (detailed in the following):

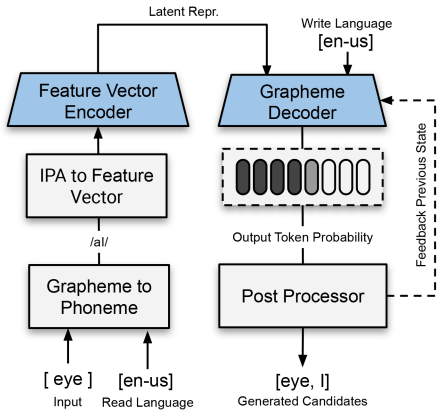
(1) The ***Grapheme to Phoneme*** (G2P) component converts the input word presented in grapheme format (in the example, the word "eye") into its corresponding representation in the IPA, given a read language (such as /aɪ/ for British English).
(2) The ***IPA to Feature Vector*** component processes each segment of the IPA representation, substituting it with a feature vector that aligns with acoustic attributes.
(3) The ***Feature Vector Encoder*** component transforms the sequence of feature vectors into representation in a latent space using a Transformer-based sequence-to-sequence model.
(4) The ***Grapheme Decoder*** (P2G) component interactively decodes the latent representation into characters to compose a new grapheme form that presents similar pronunciation of the input word (such as "I"). The initial state fed to the model selects the target language.
(5) The ***Post Processor*** component engages in beam search across the output logits and makes token selections according to probabilities derived from the decoder's output. This process is key to generating numerous quasi-homophones from a single pronunciation.

In essence, the model acts as a function F(target word, read language, write language), where the "target word" is the word for which we want to create sound-squatting candidates. The "read language" represents the language used to pronounce the word, as the same sequence of letters can have different pronunciations in different languages. Similarly, the "write language" denotes the language in which we aim to transcribe the pronunciation (i.e., the language of the sound-squatting victim). The output of this function is a set of sound-squatting candidates that belong to the target language and include homophones, quasi-homophones, and so forth.

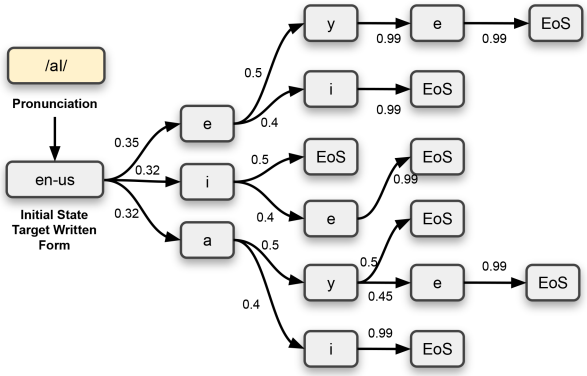
Next, we take a closer look at each component, breaking down their roles and contributions.

3.1 *Grapheme to Phoneme* (G2P)

The *Grapheme to Phoneme* component transforms written words into their respective IPA representations for a specific language. There are different tools available for that, including solutions that work for multiple languages. The most common approaches are either (1) rule based or (2)



(a) Architecture used during inference. The process to generate candidates comprises an *Feature Vector Encoder* that maps the input to a latent space and a *Grapheme Decoder* that reconstructs the input. In blue, there is the Transformer.



(b) Illustration of the inference process with $p = 0.8$. At each inference step, we explore n next characters whose probabilities add up to p. For readability, we round probabilities in the figure.

Fig. 2. Illustration of the inference process of X-squatter.

data driven. Rule-based models employ predefined rules to convert word pronunciations based on their spelling. Data-driven models use machine learning techniques and annotated data to build a model for this task.

X-squatter can use any G2P model. Here, we chose to use the eSpeak NG (Next Generation) text-to-speech engine [11] and Epitran [28] for transliterating text into IPA. eSpeak NG has better performance for English-GB and English-US. Epitran is more generic and suitable for other languages.

3.2 *IPA to Feature Vector*

The *IPA to Feature Vector* module maps IPA segments to respective *Articulatory Feature Vectors*. The technique produces a rich representation of IPA segments, introducing a concept of similarity that is absent in pure IPA representation. Being IPA is a symbolic alphabet, it is not possible to

measure how similar two IPA segments are, e.g., in terms of the sound of their pronunciations. This understanding of similarity becomes crucial for cross-language sound-squatting generation, as each language may include only a subset of IPA segments. Given two differing languages, it thus becomes necessary to search within the destination language set for other phonemes that can effectively replace those from the original language. Equally, the definition of a metric that quantifies the similarity between phonemes allows us to search for quasi-homophones and words with similar pronunciations, better controlling the quality of the generated candidates.

Projects like [10, 25, 27] aggregate IPA segments and associated features for different languages. Several proposals exist for creating Articulatory Feature Vectors that encompass features shared across languages. Implementations such as in [21, 29, 39] convert IPA segments into these vectors, which are then employed in various **Natural Language Processing (NLP)** applications. We use PanPhon [29], one of the most recent proposals, which maps over 6,000 IPA segments to 21 sub-segmental articulatory features.

3.3 *Feature Vector Encoder* and *Grapheme Decoder*

The fundamental components of X-squatter are the *Feature Vector Encoder* and *Grapheme Decoder*. They are Transformer Neural Networks [47]. These transformers rely on the self-attention mechanism to learn how to translate a sequence of feature vectors to grapheme format. Readers unfamiliar with the Transformer architecture can find more details in Appendix A.

We rely on labeled data and standard algorithms employed for training Transformer Neural Networks in a **sequence-to-sequence (Seq2Seq)** task. A Seq2Seq task is a type of machine learning problem where the goal is to transform an input sequence of data into an output sequence of data, often of different lengths and structures. Seq2Seq models typically consist of two main components: (1) an encoder, which processes the input sequence and encodes it into a fixed-length context vector (the latent representation), and (2) a decoder, which generates the output sequence based on the context vector and previously generated tokens. The Transformer is an auto-regressive model at inference; i.e., it can leverage its history of predictions to forecast future states.

When presented with a sequence of feature vectors that are obtained from the *IPA to Feature Vector* model and predictions for each of the preceding $(N-1)$ characters, the *Grapheme Decoder* module estimates the probabilities associated with each potential character becoming the Nth character of the output. Subsequently, the *Post Processor* component comes into play, analyzing these probabilities and feeding the historical context back to the *Grapheme Decoder* for generating the next forecast.

Our training approach involves instructing the transformers to take a feature vector input and generate (1) the corresponding ISO code of the language and (2) the word's written pronunciation form. For example, given the feature vector of the input "/ˈwɑtəɹ/," the correct output sequence is "en-us water." Similarly, if provided with "/ˈwɔːtə/," the output should be "en-gb water." When faced with such tasks, our model assimilates the associations between certain combinations of IPA segments and specific languages or accents. We design X-squatter as a multi-language model, with the explicit capability of generating cross-language homophones and quasi-homophones. During inference, we control the language used to read the grapheme by changing the language or accent of the G2P model.

In the cross-language case, for example, suppose we pronounce "water" in English-US and specify that we want the grapheme form to be transliterated into French-FR. In that case, the model can generate the quasi-homophone "warères" (/waʀəʀ/), which does not exist as a word in French-FR but has a similar pronunciation to "water" in English-US.

To specify the language the transformer shall use to transliterate the phoneme back into grapheme form, we provide as the start of the decoder input the target language ISO code.

3.4 *Post Processor*: The Quasi-homophone Generation

The *Post Processor* is the last element of the inference: it receives as input the *Grapheme Decoder*'s probabilistic forecasts of the next character, and it keeps track of the history of predictions to feed back to *Grapheme Decoder*. Because *Grapheme Decoder* operates as an auto-regressive model at inference, we can generate more than a candidate quasi-homophone by providing different sequences of previous $N-1$ characters. We use a Beam Search to pick from among the tokens those whose probabilities add up to p, also known as `top-p` strategy.

Figure 2(b) shows the exact output for four iterations. At each step, the *Post Processor* stores the C most likely predictions of *Grapheme Decoder* whose probabilities add up to at least p and constructs alternative histories for the next step. Figure 2(b) shows this process with a directed graph diagram (with $p = 0.8$) starting from the IPA representation of `eye`. After four iterations, the process generates six ways to write `eye`. Each branch stops when *Grapheme Decoder* outputs the special character `EoS`.

The number of iterations (M), maximum number of candidate predictions (K), probability (p), and temperature (t) are parameters that we can define manually.[4] For this work, we empirically define $M = N + 6$ times, where N is the size of the source word. We discuss the choice for p and t in Section 4.

4 X-SQUATTER: TRAINING AND VALIDATION

We now detail the training procedure and validation of X-squatter. We train our model using four distinct languages: English-US, English-GB, French-FR, and Portuguese-BR. The selection of English is grounded in its ubiquitous usage on the Internet. The inclusion of both English variants serves to highlight the influence of accents in homophone-based impersonation. Additionally, we opt for two Latin languages: French-FR, which lacks phonetic regularity, making it more susceptible to transliteration confusions, and Portuguese-BR, a more phonetically consistent language where the alignment between graphemes and phonemes is uniform. Moreover, we have restricted ourselves to those languages since we managed to present results to speakers who are fluent in at least two of the selected languages, thus helping us to get a qualitative validation of results. Extending the analysis to other languages, in particular out of the proto-Indo-European spectrum, is left for future work.

4.1 Dataset and Training

The training dataset consists of the list of English-US, English-GB, French-FR, and Portuguese-BR words from the GNU Aspell [6] word list. GNU Aspell is a free and open source spell checker containing word lists for multiple languages. To acquire the pronunciation, we use rule-based G2P tools: eSpeak NG for English-GB and English-US and Epitran for French-FR and Portuguese-BR.

The training involves taking an IPA input and generating the corresponding ISO code for the associated language, together with the word in written form. The training process of X-squatter is depicted in Figure 3. Dashed lines delineate the perimeter of the learnable parameters. There are two important differences in the training process when comparing X-squatter to the other models presented in Figure 1: (1) recalling that X-squatter is a hybrid solution, note how the Transformers expect features that encode audio properties, and (2) the target language is passed directly to the *Grapheme Decoder* block and, as such, should not be encoded in the latent features.

The training process involves a sequence-to-sequence task. The input sequence consists of a text and the "Read Language" ISO code to produce the pronunciation. The *Grapheme to Phoneme*

[4]The temperature is a normalization factor that allows the model to be more or less innovative in the generation.

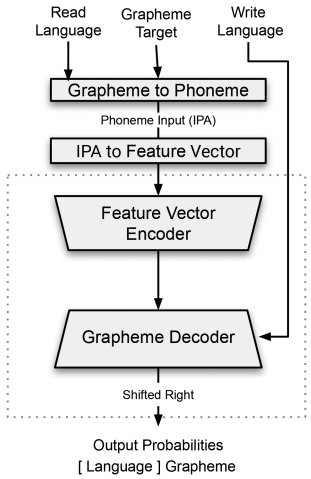


Fig. 3. X-squatter training process.

Table 3. Dataset Size for Each Chosen Language

Language Tag	Language	Region	Data Size
en-GB	English	United Kingdom	65,118
en-US	English	United States	125,923
fr-FR	French	France	245,971
pt-BR	Portuguese	Brazil	95,943
	Total		532,955

module processes the input text and generates a sequence of IPA tokens based on the "Read Language," which defines the pronunciation of the input in that language.

These IPA tokens are then passed to the *IPA to Feature Vector* module, where each token is transformed into a vector representing the acoustic features of pronunciation. The *Feature Vector Encoder* uses the Transformer's attention mechanism to map these vectors to an embedding space. In the final stage, the *Grapheme Decoder* generates the output token sequence, taking as input the "Write Language" ISO code. During training, this ISO code should match the "Read Language" ISO code, and the sequence should also contain the correct spelling of the pronunciation. The loss is calculated based on the expected output, which consists of the sequence of ISO codes plus the Grapheme Target.

Table 3 shows the size of the dataset for each language. In total, we use 437,012 words and their pronunciation. We train the *Feature Vector Encoder* and the *Grapheme Decoder* using a batch size of 16 words. Our training dataset comprises 80% of the samples, while 10% is allocated for validation and another 10% for the test set. The maximum sequence length is limited to 50 tokens. Following the standard practices on machine learning, we employ the validation set for model selection, while the test set serves to assess over-fitting. We set a maximum of 100 epochs for training, setting an early stop heuristic to interrupt the training in case of no progress in the validation set for three consecutive epochs. Indeed, training has converged after 38 epochs (about $1.15M$ steps), which took around 12 hours on a single Nvidia Tesla v100.

Table 4 lists the number of trainable parameters we used to train X-squatter and the baselines. This aspect affects resource consumption and training time. Since all models use the same base

Table 4. Model Size of the Different Alternative Approaches

Model	# Trainable Parameters
X-squatter	23.8 M
Baseline 1	22.1 M
Baseline 2	60.2 M
Baseline 3	22.2 M

Table 5. Examples of Homophones Obtained from Their IPA Representations

IPA Pronunciation	Homophones
waɪt	white, wight
slaɪt	sleight, slight
ɜːn	earn, urn
bɔl	ball, bawl
neɪ	nay, neigh

architecture (Transformer), the number of parameters has the most influence on resource consumption, i.e., GPU memory. In terms of size, all models are similar, with an exception for Baseline 2. The reason for that is the inclusion of audio signals as input. The need for reconstructing the audio signal via Mel Spectrogram requires the addition of other modules in the architecture of the tool. The new module is responsible for the 38-million-parameter difference in the size of Baseline 2.

4.2 Model Validation

The rationale behind our validation lies in confirming whether X-squatter can generate well-accepted written forms for any specific pronunciation. For that, we assess the model coverage of known homophones given a target word. The better the coverage of known homophones of a model when generating candidates, the better its generation capacity.

Our verification is divided into two parts. First, we verify the average coverage within a single language, wherein we chose English-US. In the second part, we focus on cross-language homophones. An instance of cross-language homophones is illustrated by:

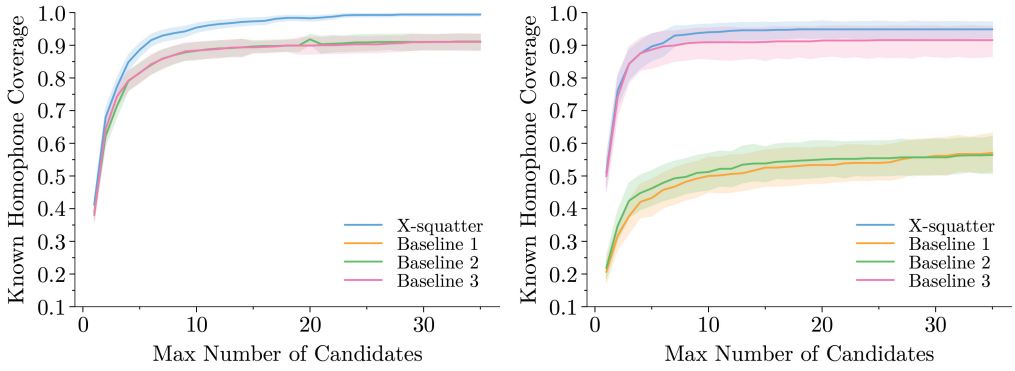
— **Hache** (French-FR) [aˈʃ]: In French, *hache* means "axe."
— **Ash** (English-US) [aˈʃ]: In English, *ash* refers to the residue from burning.

The two words—from distinct languages—are cross-language homophones since their pronunciations (represented in IPA) are identical. Often, the involved words assume very different written representations and meanings in these cases. Yet, they are good sound-squatting candidates, i.e., in this example, the victim could be a French person who must write **ash** heard with proper English pronunciation.

4.3 Single-language Homophone Coverage

We first evaluate the single-language scenario. For that, we obtain a complete list of homophones for English, using the curated list of known homophones provided by the AIL tool [26, 32]. Some examples of homophones are provided in Table 5.

This list contains 362 pronunciations in English-US represented as IPA transcripts. Each IPA transcript is associated with at least two English words, which are thus homophones. This curated list serves as our ground truth. Our evaluation involves inputting the IPA transcripts into both



(a) Single-language (b) Multi-language

Fig. 4. Homophone coverage for single- and cross-language scenarios: As the parameter K (maximum number of candidates prediction) increases, the model exhibits a higher coverage.

the baseline models and X-squatter and gathering the candidates generated by each model. When making predictions with the models, we set the beam search with a probability threshold of $p = 0.9999$ and a temperature value of $t = 1.0$.

We then measure the percentage of known homophones generated by each model when producing a given number of candidates for each of the 362 pronunciations. Clearly, the more candidates we produce, the higher the probability of generating all known homophones. We expect good models to generate the known homophones earlier since known homophones are the best candidates for sound-squatting.

Results are summarized in Figure 4(a), which depicts the average coverage with lines. Results for the three baseline models are mostly coincident, and thus lines overlap in the figure. The color ranges mark the confidence intervals. As expected, increasing the maximum number of candidates per pronunciation systematically increases coverage. While the baseline models arrive at a maximum coverage of 90% of the homophones with 30 candidates per pronunciation, X-squatter generates *all* homophones already with less than 20 candidates per pronunciation.

4.4 Multi- and Cross-language Homophone Coverage

We complement the analysis with the multi-language scenario. As there is no curated list of cross-language homophones, we collect all pronunciations from the training dataset and mark as cross-language homophones the words showing exactly the same pronunciation across different languages.[5] Recall that models are trained using phoneme/grapheme pairs always of the same language. As such, models have not seen cross-language homophones during training.

We find a total of 95 pronunciations with multiple written forms in various languages. Notably, since we have the rigid requirement that the pronunciation must be identical across at least two languages, the number of cross-language homophones is relatively small. In total, we observe 374 written forms spanning four languages. Examples of cross-language homophones are shown in Table 6.

Following the same approach as for the single-language validation, we input these pronunciations into both the baseline models and X-squatter, capturing the candidates generated by each model. Recall that the model can be represented as a function

[5]We engaged multi-language speakers in the used languages to validate these lists of cross-language homophones.

Table 6. Examples of Cross-language Homophones with IPA Pronunciations

IPA Phoneme	Homophones
ɛʃ	$[fr-fr]$ ais, $[en-us]$ esse
ʃɪ	$[fr-fr]$ chie, $[pt-br]$ xi, $[en-us]$ shi
kædi	$[fr-fr]$ caddie, $[pt-br]$ cádi, $[en-gb]$ caddie, $[en-gb]$ caddy
maʃ	$[fr-fr]$ mâchai, $[pt-br]$ más, $[en-gb]$ mache, $[en-gb]$ mash
kaʃ	$[fr-fr]$ cache, $[pt-br]$ chás, $[en-gb]$ kasch

F(target word, read language, write language). In the cross-language case, we modify the input to F(unknown word, unknown language, write language). By varying the "write language," we generate the output for all four target languages, aggregating the outcomes. During predictions, we again utilize a probability threshold $p = 0.9999$ and a temperature value $t = 1.0$.

Figure 4(b) provides a visualization of the results. Similar to the single-language scenario, increasing the maximum number of candidates systematically enhances the average coverage. However, we observe a divergence among the baseline models. As expected, two baseline models do not support multi-language scenarios and as such achieve a maximum coverage of 60%. That is, they identify only the cases where there are also intra-language homophones in the dataset.

X-squatter stands out, generating around 94.89% of the known cross-language homophones with less than 10 candidates per pronunciation. These results confirm that X-squatter candidates are of good quality, including all single-language homophones and most of the cross-language ones already in its first few candidates. X-squatter is, however, able to generate many more candidates, including quasi-homophones and words with similar pronunciation, i.e., cases that are completely neglected by the list-based methods used for producing our validation ground truth. In the following, we show how X-squatter's candidates can be used for the proactive search of sound-squatting abuses.

5 SEARCHING FOR SOUND-SQUATTING

We now apply X-squatter in the search for sound-squatting in practical use cases. Specifically, we compile a collection of popular names, derive their homophones and quasi-homophones across multiple languages, and actively check whether (1) they have been used in practice and (2) they may be sound-squatting abuses.

We focus on two types of resources: PyPI packages (Section 5.1) and domain names found in registered TLS certificates (Section 5.2). In both cases, when generating sound-squatting, we also use alternative squatting tools for generating candidates, comparing the results of our analysis of sound-squatting against other types of squatting. We use as a baseline the AIL tool [32], generating all candidates for each given target. This tool incorporates a set of 21 algorithms encompassing typo-squatting, homograph-squatting, and list-based sound-squatting (see Section 2). For simplicity, we will refer to the candidates generated by AIL as "other-squatting."

By contrasting our sound-squatting approach with the other-squatting techniques generated by the AIL tool, we show that our data-driven approach can uncover threats neglected by the baseline, leaving resources exposed to attacks.

5.1 Squatting on PyPI Repository

We first investigate the squatting opportunities on PyPI. Here we limit ourselves to finding packages in the PyPI platform that are good squatting candidates for popular packages. We study the characteristics of these packages and their statistics, comparing sound-squatting against our baseline. Verifying the actual maliciousness of all these candidates is out of our scope—instead,

Table 7. Generated Candidates and Projects Found Online per Technique, Along with the Intersection Comprising Candidates Matching Both Sound-squatting and Other-squatting Techniques

	Sound-squatting	Other-squatting	Intersection
Generated	518,180	7,637,268	21,870
Found	1,579	16,770	1,050

we shed light on the attack opportunities, illustrating how sound-squatting candidates differ from other-squatting.

5.1.1 Background. The **Python Enhancement Proposal (PEP)** 541 [13] is explicit in its stance against name squatting within the PyPI ecosystem. This prohibition serves as a preventive measure against the distribution of malware through the manipulation of users' potential confusion regarding package names. A stark illustration of these concerns emerged in November 2022 when Phylum[6] published an article detailing PyPI attacks: malicious actors utilized Python to engineer a JavaScript extension that substitutes cryptocurrency addresses in clipboards with wallet addresses under the attackers' control. Generally, documented attacks against the PyPI ecosystem primarily rely on typo-squatting. Yet, there has been anecdotal evidence of squatting that unintentionally affected non-English speakers.[7]

These incidents underscore the intrinsic risks associated with name squatting within platforms such as PyPI. Sound-squatting may introduce greater complexities for moderators to detect, potentially allowing packages to evade surveillance.[8]

5.1.2 Methodology. We start from a list of 5,000 highly popular packages from the PyPI repository based on their download counts in the last 30 days before June 20, 2023. Subsequently, we generate candidate package names using both sound-squatting and other-squatting techniques. Next, we search the repository index to determine the occurrence of these candidate packages. Our analysis focuses on tracking the existence of these packages over 967 days, from November 20, 2020, to July 14, 2023.

In total, we generate a substantial number of candidate package names. Specifically, we create 518,180 sound-squatting candidates and 7,637,268 other-squatting candidates across the entire set of 5,000 packages. That is, there are approximately 15 times more other-squatting candidates per project, which is expected given the diverse range of algorithms utilized for other-squatting generation.

5.1.3 PyPI Package Candidates. Table 7 shows the number of candidates evaluated and the total number of candidates that exist on the platform at some point in the evaluation period. We also show the intersection between the techniques to illustrate that sound-squatting is not covered by the other algorithms.

As said above, the number of other-squatting candidates is around 15 times the number of sound-squatting candidates. Yet, there is a first interesting difference in the proportion of candidates found to exist on the platform. While ≈2.2 packages are found online for every 1,000 candidates for other-squatting, this ratio increases to ≈3.0 for sound-squatting. It is also interesting to check

[6] https://blog.phylum.io/phylum-discovers-another-attack-on-pypi/
[7] https://metrodore.fr/i-have-been-powned.html
[8] Notice that we do not envision only scenarios where users rely on voice commands to install packets, which may be rather rare currently. Instead, we believe a greater threat in this case is represented by users who must type in the name of packets heard from others.

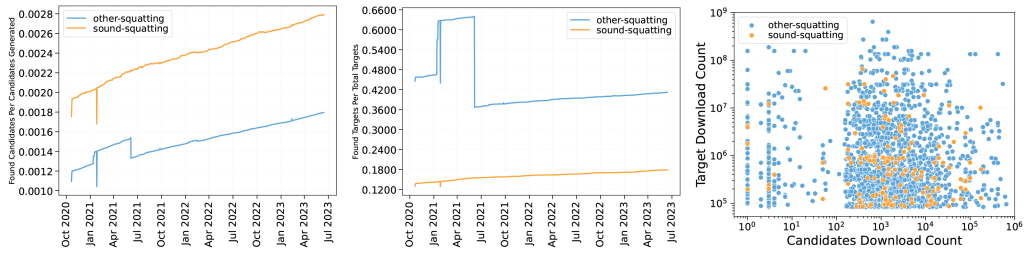


Fig. 5. Temporal analysis of squatting candidates in PyPI. (a) Found candidates over total generated candidates. (b) Found candidates over the number of evaluated targets. (c) Downloads of candidates vs. downloads of target packages.

the intersection of the lists. Names found in both lists have a much higher chance of being found on the platform: ≈48 for every 1,000 candidates is indeed found online. That is, X-squatter offers a good method to refine and prioritize the search for squatting in lists generated by the other tools.

This behavior is consistent over time, as depicted in Figure 5(a). The figure shows the cumulative count of found candidates per day, divided by the total number of generated candidates. Separated lines show sound-squatting candidates (yellow) and other-squatting (blue). The share of sound-squatting candidates found online is consistently higher than other-squatting, growing almost linearly over time as new packages are added to PyPI.

In the temporal analysis, we note a particular behavior starting in February 2021. To improve its analysis, we show the share of packages found online per day normalized by the number of targets (5,000) in Figure 5(b). Naturally, other-squatting is above sound-squatting in this figure as we have more other-squatting candidates than sound-squatting candidates.

On February 12, 2021, a total of 573 packages from our list of typo-squatting candidates were created in the platform. We see in Figure 5(b) a major increase in the share of other-squatting candidates found online, followed by a sharp drop. Between February 25, 2021, and February 26, 2021, approximately 73,000 packages faced suspension in PyPI due to violations of the platform rules. In the aftermath, on February 27, a large portion of the suspended packages were reinstated (see the plateau after that day in Figure 5(b)), as not all removed packages could be linked to malicious activity. Finally, on July 5, 2021, 1,610 packages were permanently deleted from PyPI, with a substantial number also present in our generated dataset, as shown by the sharp drop for other-squatting in Figure 5(b). This event has been documented in reputable sources.[9]

Interestingly, sound-squatting candidates have mostly been ignored in the February 2021 incident. Yet, package statistics suggest that it does represent a threat that could be equally exploited.

Figure 5(c) displays a scatter plot of the number of downloads of the package targets (y-axis) versus the number of downloads of the candidates found online (x-axis). Notice the different scales. Each point marks a candidate and colors represent sound-squatting (yellow) and other-squatting (blue). Sound-squatting candidates are found in the complete spectrum of the figure, with download statistics comparable to other-squatting types. Notice in particular that many sound-squatting candidates present large download counts.

We finally perform a manual qualitative analysis of some sound-squatting candidates generated by X-squatter, presenting in Table 8 salient examples. Some of these packages are periodically deleted by the platform and republished right after the removal. Not all of them are malicious. In

[9]See https://www.theregister.com/2021/03/02/python_pypi_purges/ and https://github.com/pypi/support/issues/935

Table 8. Examples of Candidates Generated by X-squatter and Found Online in the PyPI Repository

Squatting	Legitimate Package	Context
scrapi	Scrapy	Empty repository
sfinx	sphinx	Low-reputation project without documentation
requestes	requests	Warning page
regex	rejex	Affiliated project (legitimate)
skema	schema	Low-reputation project without documentation
flasque	flask	Low-reputation project without documentation
noompy	numpy	Project uses similarity as a form of homage
pidantic	pydantic	Low-reputation project without documentation
pirec	pyrect	Unrelated. Squatting project, allow shell command execution

particular, `requestes` is a package that warns users against installing packages without checking them first. Most packages have a low reputation with almost no documentation, which indicates a parking or squatting attempt. One striking example is `pirec`, which is very similar in pronunciation to the legitimate package `pyrect`. However, `pirec` is a software that can execute shell scripts, giving attackers a high degree of freedom once victims install the package.

These results indicate that sound-squatting represents a feasible technique for malicious actors seeking to propagate malware via PyPI. We believe X-squatter can assist PyPI's moderators in adopting measures to effectively mitigate the potential risks associated with sound-squatting.

5.2 Domain Impersonation

We now move into domain squatting, specifically within the context of HTTPS domains. Squatting in domains with valid TLS certificates is a serious threat as valid certificates can fool users into believing the squatted domain is legitimate. As for the previous use case, we focus on comparing sound-squatting to other types of squatting. While some examples of malicious sound-squatted domains are provided, a comprehensive verification of maliciousness is beyond the scope of our work.

5.2.1 Methodology. Attackers are known to register TLS certificates for squatted domains to increase the success rate of phishing campaigns. We thus search for squatting candidates using a dataset of registered domains found in certificates collected via CertStream (Certificate Transparency Logs).[10] We select the 10,000 most accessed domains from the Tranco Ranking List [31] (accessed on July 10, 2023).

Using X-squatter, we generate a total of 545,509 unique names from these 10,000 candidates, employing several cross-language configurations. In this use case, we set X-squatter parameters to limit the number of candidates to 35 per domain, and use as a threshold $p = 0.999$ and a temperature of $t = 1.0$. Again, we generate candidates for other-squatting candidates with AIL Typo-Squatting. We obtain 7,853,273 other-squatting candidates, and, among those, 52,089 candidates fall within the intersection of the two categories.

Using the CertStream data, we extract all server names to build a list of registered second-level domains. By scrutinizing these names, we identify domains that have the potential to impersonate popular websites, even in their nascent stages. This early identification is pivotal in preventing squatting attacks. We collected certificates spanning 124 days, from February 11, 2023, to June 14, 2023. In total, we gathered 866,125,758 certificates representing 127,775,910 domains across 5,155 **Top-Level Domains (TLDs)**. These certificates originated from a total of 263 certificate issuers

[10] https://certstream.calidog.io/

Table 9. Candidates Generated and Registered Domains per Technique, Along with the Intersection of Candidates Matching Both Sound-squatting and Other-squatting Techniques

	Sound-squatting	Other-squatting	Intersection
Generated	545,509	7,853,273	52,089
Found Registered	84,377	607,010	32,490

located in 50 different countries. Processing this longitudinal dataset required substantial resources (including a PySpark cluster). It is worth noting that conducting daily searches is also a feasible approach with few resources.

5.2.2 Registered TLS Certificates. Table 9 details the total number of candidates and those we found in CertStream. We notice a trend similar to the PyPI use case, however, with much higher percentages. While ≈7.7% of the candidate TLS certificates are found online for other-squatting, this ratio increases to ≈15.5% for sound-squatting. Names found in both lists again have a much higher chance of being found on CertStream: ≈62.4% are indeed online. This reinforces the importance of checking sound-squatting and the use of X-squatter for proactive defense.

The percentages of existing candidates are striking for both other-squatting and sound-squatting. To observe how the phenomenon evolves over time, Figure 6(a) presents the proportion of registered candidates we identified as existing, compared to the total number of domains found in all certificates issued on each respective day. Notice how the ratio is constant over time, with some minor spikes in particular days. That is, the registration of potentially abused domains is very high and continuous. Naturally, other-squatting is above sound-squatting as we have much more other-squatting candidates.

Figure 6(b) provides insight into the relationship between TLDs and sound-squatting candidates. Notably, the TLD `.io` stands out with an average of 467.46 (standard deviation 146.94) unique registered domains that match our candidates per day, a share of 3.98%. To complement the picture, in Figure 6(c), we represent the position of the most popular TLDs according to the number of registered domains and their ranking position based on the number of squatting candidates per TLD. Ideally, we would expect a linear relationship around the red line, indicating that the number of squatting candidates is proportional to the number of domains managed by each TLD. This is indeed the general trend. However, we also observe some outliers in the figure.

Among the top 100 most present domains, we find that TLDs such as `.io`, `.app`, and `.dev` deviate significantly from the diagonal line. Even more concerning is the outlier TLD `.us`. These findings align with recent reports of phishing campaigns hosted in the `.us` TLD.[11]

This variation among TLDs in their susceptibility to squatting activities highlights the need for a nuanced examination of each TLD's security and potential misuse.

Finally, Table 10 summarizes the number of candidate domains found online, grouped by the type of sound-squatting they represented. Some domains are potentially abused in a unique cross-language combination since the global unique abused domains are 9,586. This provides evidence that sound-squatting also considers cross-language homophones. Considering other-squatting, we find 607,010 names with the potential of being abused for 3,661,705 unique registered domains, which represents a total of 8.98% of the generated candidates. In total, 8,878 domains can be potentially squatted by at least one candidate using the typo-squatting technique.

Also in Table 10, one of the languages used in the Phoneme to Grapheme (target language) is Spanish-ES. This language is not on the training data; however, X-squatter is able to model

[11] https://it.slashdot.org/story/23/09/02/1415234/why-are-godaddys-us-domains-being-used-for-so-much-phishing

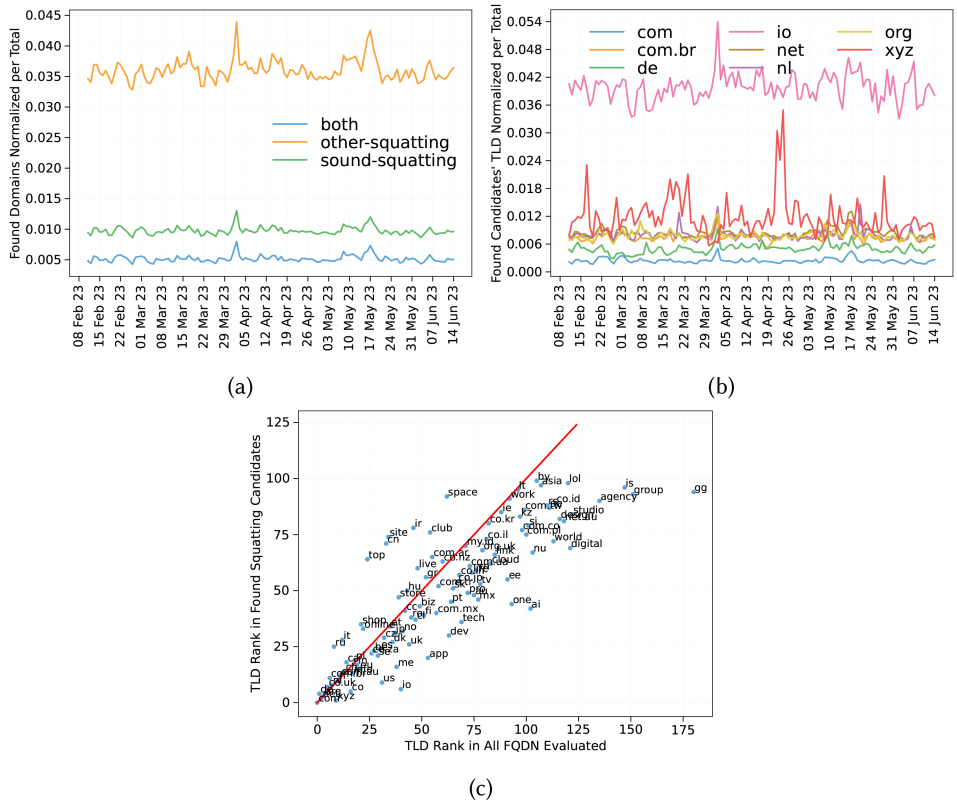


Fig. 6. Temporal analysis of squatting candidates in SSL/TLS certificates collected from Certificate Transparency: (a) Registered candidates over total registered domains on issued certificates per day. (b) Ratio of registered sound-squatting candidates per TLD. (c) Relationship of TLD and squatting candidates.

Table 10. Results for Cross-language Sound-squatting by Language Combination

	Sound-squatting				Other-squatting
	G2P				
P2G	en-GB	en-US	fr-FR	pt-BR	
en-GB	0.791 / 21,606 / 0.0034	0.750 / 20,597 / 0.0032	0.649 / 16,521 / 0.0026	0.681 / 19,076 / 0.0030	
en-US	0.754 / 20,467 / 0.0031	0.795 / 21,394 / 0.0033	0.656 / 16,592 / 0.0026	0.700 / 19,401 / 0.0030	
es-ES	0.578 / 15,096 / 0.0023	0.601 / 15,954 / 0.0024	0.425 / 9,876 / 0.0016	0.575 / 12,640 / 0.0021	
fr-FR	0.677 / 19,557 / 0.0031	0.675 / 19,550 / 0.0031	0.636 / 15,086 / 0.0026	0.639 / 15,814 / 0.0026	
pt-BR	0.650 / 16,117 / 0.0023	0.668 / 16,768 / 0.0025	0.431 / 9,989 / 0.0016	0.714 / 15,329 / 0.0025	
Overall	9,586 / 84,377 / 1,116,227				8,878 / 607,010 / 3,661,705

The table displays separated by a '/' (i) the ratio of targets with at least one existing candidate, (ii) the number of registered candidates, and (ii) the ratio of registered candidates over the total number of registered domains analyzed. Note that a single candidate might be registered across multiple top-level domains (TLDs), leading to a higher count of registered domains than candidates.

the language and produce results similar to the other languages that are present in the training set. This result suggests that the representation power of X-squatter can be generalized to other languages, which we will investigate further in future work.

5.2.3 Manual Qualitative Analysis. To assess the effectiveness of our generation, we chose domains with known histories of phishing attacks. Specifically, we choose amazon, bankofamerica,

dropbox, facebook, icloud, instagram, linkedin, microsoft, netflix, paypal, steamcommunity, and tripadvisor for our analysis.

Subsequently, we obtain the candidates using X-squatter and conduct a manual verification process to ascertain whether these domains exhibited phishing characteristics. The manual verification covers 1,038 domains. We use the tool Puppeteer to capture screenshots, setting a timeout of 2 seconds.

We organize the candidates into nine distinct classes that effectively capture legitimate/abuse characteristics. This classification scheme aids in gaining insights into the nature and potential purposes behind these domains:

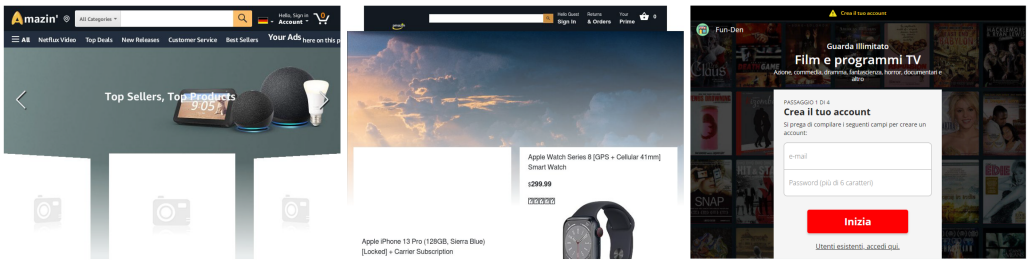
- **Error Domains (463 domains):** The "Error Domains" category consists of domains that do not present any content and are shown as errors in the browser. These domains likely represent typographical errors, misconfigurations, or inactive websites. While they may not be malicious, they still have relevance.
- **Legitimate Domains (279):** These are genuine, non-squatting domains that are not involved in any malicious or deceptive activities. They serve their intended purpose without any evidence of fraud.
- **Domains for Sale (117):** Domains in this category are put up for sale. While not necessarily malicious, they might be squatting on potentially valuable domain names with the intention of selling them at a higher price.
- **Parked Domains (90):** Parked domains are placeholders typically used by domain owners or registrars. They often display advertisements or a generic landing page, and their primary purpose is to generate revenue through ad clicks.
- **Redirector Domains (40):** Redirector domains are used to redirect web traffic from one domain to another. They can be legitimate, but they may also be used in various online scams and malicious campaigns.
- **Hit-stealing Domains (16):** Hit-stealing domains might be involved in schemes where they try to intercept or steal web traffic intended for other legitimate domains, potentially for fraudulent purposes.
- **Look-alike Domains (10):** Look-alike domains often resemble well-known, legitimate pages, but with slight variations that might go unnoticed. They are typically used in phishing or deception attempts.
- **Phishing Campaign (7):** Domains categorized under this label are likely part of phishing campaigns, where they are used to trick users into revealing sensitive information or credentials.
- **Other (16):** This category includes domains that do not fit neatly into the previously defined categories or require further investigation to determine their purpose and intent.

We show in Figure 7 some examples of phishing domains found during the classification.

Interestingly, some brands chose to buy problematic domains and redirect traffic to their services. We call these Authoritative Owned domains. Focusing on the Redirector Domains, we discovered by checking DNS data that 18 quasi-homophone domains for Amazon and Apple are Authoritative Owned. Four domains are owned by an organization responsible for brand protection. Eight domains belong to third-party organizations, and for 11 domains we could not find any information regarding ownership. After this inspection, we conclude that measures taken by high-profile organizations against domain impersonation include some sound-squatting candidates. However, there are many missing candidates, which thus represents an opportunity for attackers.

6 DISCUSSION AND LIMITATIONS

We now discuss limitations related to both our proposed tool and experiments.



(a) Phishing Amazon (b) Phishing Amazon (c) Phishing Netflix

Fig. 7. Phishing examples for Amazon.com and Netflix.com found in selected candidate domains.

6.1 Coverage

Language Scope. Our investigation focused on languages within the Indo-European family, specifically English (United States and Great Britain), French (France), Portuguese (Brazil), Italian (Italy), and Spanish (Spain). These languages collectively represent a substantial portion of online users and resources [33]. While this choice allows for an extensive evaluation of phonetic aspects due to the availability of tools and resources, it excludes important languages from other language families such as Russian, Chinese, and Arabic.

X-squatter can be extended to cover these languages. The IPA applies universally, enabling us to obtain Feature Vectors for other languages. We have illustrated this flexibility by reporting results for es-ES (Spanish from Spain) as the input language in Table 10, even though the model has not been trained with this language. The application of X-squatter for other languages requires that *Grapheme to Phoneme* be adapted to the input language, while a specific *Grapheme Decoder* module must be adjusted to specific written forms of the output language via fine-tuning or retraining.

Also, regarding adaptability, a potential challenge arises if the rules of *Grapheme to Phoneme* are no longer applicable, which would be the case if new words are added to a language, e.g., due to the natural evolution of languages. However, languages often adapt spellings and pronunciations slowly, making this scenario rare. Moreover, words borrowed from one language to another typically maintain a similar pronunciation but change in written form, and thus are in line with our assumptions.

Handling Heteronyms. A limitation in our approach is its lack of handling of heteronyms, i.e., words with the same written form but different pronunciations, common in non-phonetic languages like English and French. Again, this is mostly a problem to be addressed in the *Grapheme to Phoneme* module. The eSpeack NG is designed to produce a single pronunciation alternative and does not cover heteronyms. Properly handling them requires using a *Grapheme to Phoneme* that generates all alternative pronunciations for the same word. Measuring the practical impact of the lack of this feature is challenging, as their classification depends on context.

6.2 Applicability

We do not provide a comprehensive measurement study that quantifies sound-squatting actual abuses and their impact on end-users. Quantifying the effectiveness of attacks based on sound-squatting will require a study with real users.

Unlike typo-squatting, sound-squatting candidates may appear familiar to users, making them harder to detect. Indeed, our qualitative analyses indicate that X-squatter can be applied to improve

the search for other squatting techniques. Moreover, as mentioned earlier, we believe the threat is far from limited to voice commands. We hypothesize, for example, potential abuses exploiting misunderstanding on recommendations. These cases may occur when someone recommends products (e.g., sites or software) with foreign names without spelling. This is the case of a user listening to radio or podcast advertisements and trying to access resources for which the pronunciation may be misleading. In summary, we envision spoken language as the attack vector, extending beyond smart speakers or voice-to-text engines. However, we do not validate this hypothesis here, as it will require a different methodology—which necessarily will involve users—to collect significant datasets.

6.3 Scalability

X-squatter uses a Transformer network, which can be trained in a single-commodity server in a few hours. Compared to the list-based alternatives, X-squatter requires more computational resources during training. However, it improves the quality and coverage of the generated candidates.

While we compare different Transformer architectures during our design of X-squatter, we have not considered more recent alternatives, in particular those based on generative AI. The defining characteristic of **Large Language Models (LLMs)** is their large number of parameters [8]: for instance, Bert [9] has 100 million parameters, GPT-2 has 1.5 billion parameters [34], and LLama2 has 65 billion parameters [43]. Usually, specializing these models to specific problems requires fine-tuning, with substantial effort in terms of data collection and especially computational resources. We instead decide to build a specialized model to tackle a domain-specific problem, avoiding fine-tuning large/huge models. This allows us to reduce the model size while increasing the performance in the specific task. X-squatter has $22M$ parameters, and it leverages architectural choices fine-crafted for the homophone generation problem.

In sum, the capabilities seen in recent LLMs call for a direct comparison of X-squatter with these new models. This will require more experiments to assess not only the quality of the generated candidates but also the complexity of the complete pipeline.

7 CONCLUSION AND FUTURE WORK

This article focused on cross-language sound-squatting, a threat that has gained prominence, particularly with the rise of smart speakers and audio content like podcasts.

We introduced X-squatter, a multi-language AI-based system that leverages a Transformer Neural Network to generate high-quality sound-squatting candidates. We employed X-squatter to enumerate domain names vulnerable to sound-squatting abuse within the vast landscape of TLS certificates. We uncovered that approximately 15% of the generated candidates are associated with existing TLS certificates—a higher figure when compared to other squatting types (7%).

Additionally, we applied X-squatter to study PyPI packages, where we identified hundreds of sound-squatting candidates within a 3-year package history. We identified that sound-squatting candidates are more often neglected in periodic platform checks, as opposed to other types of squatting, calling for more proactive prevention measures.

In essence, X-squatter is a new asset for the proactive defense against the multilingual sound-squatting. It not only helps to uncover the threat but also paves the way for enhanced protection in the evolving landscape of the cyber threat.

For future work, we plan to integrate X-squatter for the protection of smart speakers. We also plan to investigate multi-modal models to include sound, together with other types of features, directly in the model training.

APPENDIX

A ARCHITECTURAL DETAILS

Figure 8(a) details the architecture of the model used for homophone generation. We include in the diagram the *Feature Vector Encoder* and *Grapheme Decoder*. We also show in Figure 8(b) the inside of the decoder block that we use to reduce the complexity of the visualization.

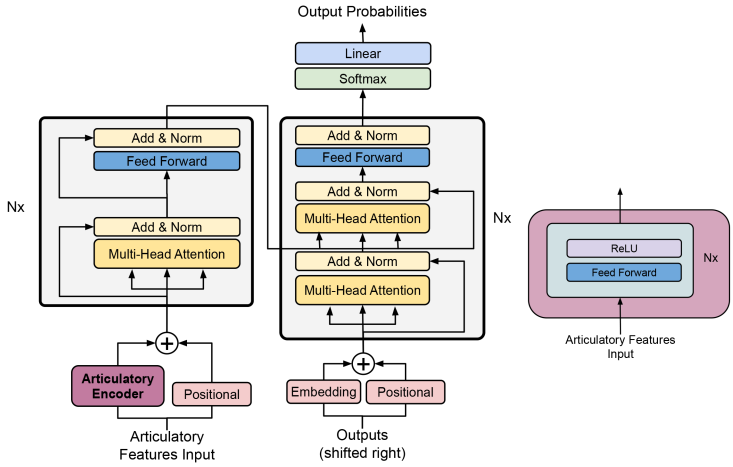


Fig. 8. (a) High-level representation of the X-squatter Transformer architecture. (b) Inside view of the Articulatory Decoder Block.

We adopted the Adam optimizer with a learning rate (LR) of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. Training halts when we observe the Validation Loss ceasing to decrease for three consecutive epochs. Symmetric hyperparameters are employed for the Transformer Network in both the encoder and decoder modules, with a hidden representation size of 512, an embedding dimension of 512, and eight attention heads. The input vocabulary encompasses 6,487 IPA segments, while the output vocabulary comprises 133 grapheme tokens, consistent across all languages. Table 11 lists the hyper-parameters.

Table 11. Compilation of Hyper-parameters for X-squatter

Hyper-parameter	Value
# Phoneme Vectors	6,487
# Grapheme Tokens	133
Phoneme Embedding Dimension	512
Feature Vector Encoder # of Attention Heads	2
Feature Vector Encoder Hidden Dimension	512
Feature Vector Encoder Linear Hidden	2,048
Feature Vector Encoder Dropout	0.1
Grapheme Decoder # of Attention Heads	2
Grapheme Decoder Hidden Dimension	512
Grapheme Decoder Linear Hidden	2,048
Grapheme Decoder Dropout	0.1
Total Number of Parameters	60.2 M

REFERENCES

[1] AIL Project. 2023. *AIL - PyPI Squatting*. Retrieved from https://github.com/typosquatter/pypi-squatting. Accessed on: 09/08/2023.

[2] Marcin Ulikowski. 2023. *dnstwister: DNS Twist Web Service*. Retrieved from https://dnstwister.report/. Accessed on: 09/08/2023.

[3] monkeym4ster. 2023. *DomainFuzz: A Domain Name Permutation Tool*. Retrieved from https://github.com/monkeym4ster/DomainFuzz. Accessed on: 09/08/2023.

[4] 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'15)*. Internet Society.

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

[6] Kevin Atkinson. 2006. *GNU Aspell 0.60.4*. Retrieved from http://aspell.net. Accessed: 09/08/2023.

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

[8] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technologies* 15, 3 (2024), 45. https://doi.org/10.1145/3641289

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv:cs.CL/1810.04805* (2019).

[10] Matthew S. Dryer and Martin Haspelmath (Eds.). 2013. *WALS Online (v2020.3)*. Zenodo. https://doi.org/10.5281/zenodo.7385533

[11] Jonathan Duddington, Martin Avison, Reece Dunn, and Valdis Vitolins. eSpeak NG Text-to-Speech. https://github.com/espeak-ng/espeak-ng

[12] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. 2013. Analysis of the HTTPS certificate ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference*. ACM. https://doi.org/10.1145/2504730.2504755

[13] Python Software Foundation. 2018. Invalid Projects. https://peps.python.org/pep-0541#invalid-projects

[14] Yacong Gu, Lingyun Ying, Yingyuan Pu, Xiao Hu, Huajun Chai, Ruimin Wang, Xing Gao, and Haixin Duan. 2023. Investigating package related security threats in software registries. In *2023 IEEE Symposium on Security and Privacy (SP'23)*. 1578–1595. https://doi.org/10.1109/SP46215.2023.10179332

[15] 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.

[16] International Organization for Standardization (ISO). 2024. ISO 3166 Country Codes. https://www.iso.org/iso-3166-country-codes.html

[17] 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.

[18] Doowon Kim, Haehyun Cho, Yonghwi Kwon, Adam Doupé, Sooel Son, Gail-Joon Ahn, and Tudor Dumitras. 2021. Security analysis on practices of certificate authorities in the HTTPS phishing ecosystem. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3433210.3453100

[19] 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.

[20] 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.

[21] Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, 8–14. https://aclanthology.org/E17-2002

[22] Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020. A Survey on Contextual Embeddings. (2020). arXiv:cs.CL/2003.07278

[23] David Lodge. 2023. URLCrazy: Domain Name Permutation and Availability Checker. Retrieved September 8, 2023 from http://www.morningstarsecurity.com/research/urlcrazy

[24] 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'20)*. 1021–1030. https://doi.org/10.1109/BigData50022.2020.9377875

[25] Susanne Maria Michaelis, Philippe Maurer, Martin Haspelmath, and Magnus Huber (Eds.). 2013. *APiCS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. https://apics-online.info/
[26] Sami Mokaddem, Gérard Wagener, and Alexandre Dulaunoy. 2018. AIL-the design and implementation of an analysis information leak framework. In *2018 IEEE International Conference on Big Data (Big Data'18)*. IEEE, 5049–5057.
[27] Steven Moran and Daniel McCloy (Eds.). 2019. *PHOIBLE 2.0*. Max Planck Institute for the Science of Human History, Jena. https://phoible.org/
[28] David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC'18)*.
[29] David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. PanPhon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING'16)*. COLING 2016 Organizing Committee, 3475–3484. https://aclanthology.org/C16-1328
[30] 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.
[31] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2018. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156* (2018).
[32] AIL Project. 2023. AIL-Typo-Squatting. https://ail-project.github.io/ail-typo-squatting/
[33] Q-Success Web-based Services. 2024. Usage Statistics of Content Languages for Websites. https://w3techs.com/technologies/overview/content_language
[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.
[35] rangertaha. 2023. URLInsane. https://github.com/ziazon/urlinsane
[36] Richard Roberts, Yaelle Goldschlag, Rachel Walter, Taejoong Chung, Alan Mislove, and Dave Levin. 2019. You are who you appear to be. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3319535.3363188
[37] Gunikhan Sonowal. 2020. A model for detecting sounds-alike phishing email contents for persons with visual impairments. In *2020 6th International Conference on e-Learning*. IEEE, 17–21.
[38] Gunikhan Sonowal and K. S. Kuppusamy. 2016. MASPHID: A model to assist screen reader users for detecting phishing sites using aural and visual similarity measures. In *Proceedings of the International Conference on Informatics and Analytics*. 6 pages. DOI: https://doi.org/10.1145/2980258.2980443
[39] Marlene Staib, Tian Huey Teh, Alexandra Torresquintero, Devang S. Ram Mohan, Lorenzo Foglianti, Raphael Lenain, and Jiameng Gao. 2020. Phonological features for 0-shot multilingual speech synthesis. *arXiv preprint arXiv:2008.04107* (2020).
[40] 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.
[41] 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 (IMC'18)*. Association for Computing Machinery, New York, NY, USA, 429–442. https://doi.org/10.1145/3278532.3278569
[42] Ivan Torroledo, Luis David Camacho, and Alejandro Correa Bahnsen. 2018. Hunting malicious TLS certificates with deep neural networks. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. ACM. https://doi.org/10.1145/3270101.3270105
[43] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[44] Rodolfo Valentim, Idilio Drago, Federico Cerutti, and Marco Mellia. 2022. AI-based sound-squatting attack made possible. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW'22)*. IEEE, 448–453.
[45] Rodolfo Valentim, Idilio Drago, Federico Cerutti, and Marco Mellia. 2023. Sound-skwatter (did you mean: Soundsquatter?) AI-powered generator for phishing prevention. *arXiv preprint* (2023).
[46] Rodolfo Valentim, Idilio Drago, Marco Mellia, and Federico Cerutti. 2023. Lost in translation: AI-based generator of cross-language sound-squatting. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroSPW'23)*. 513–520. https://doi.org/10.1109/EuroSPW59978.2023.00063
[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[48] Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski, and Brad Daniels. 2006. Strider typo-patrol: Discovery and analysis of systematic typo-squatting. In *2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'06)*. USENIX Association, San Jose, CA. Retrieved from https://www.usenix.org/conference/sruti-06/strider-typo-patrol-discovery-and-analysis-systematic-typo-squatting

[49] Yuwei Zeng, Tianning Zang, Yongzheng Zhang, Xunxun Chen, and YiPeng Wang. 2019. A comprehensive measurement study of domain-squatting abuse. In *2019 IEEE International Conference on Communications (ICC'19)*. 1–6. https://doi.org/10.1109/ICC.2019.8761388

[50] 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'19)*. IEEE, 1381–1396.

Received 11 October 2023; revised 29 March 2024; accepted 25 April 2024