The Hand-fly Effect: Investigating the Prevalence of Shift-Squatting in the Wild

Huimin Zhao*, Sydney Hester†, Xiaoqin Liang‡, Shuai Hao‡, Xing Gao†, Guannan Liu*
*Colorado School of Mines, Golden, CO, USA
†University of Delaware, Newark, DE, USA
‡Old Dominion University, Norfolk, VA, USA
{huimin_zhao, guannan.liu}@mines.edu, {sydneyph, xgao}@udel.edu, {xlian001, shao}@odu.edu,

***Abstract*—In recent decades, various forms of cybersquatting, notably typosquatting, have been extensively documented and demonstrated to be effective in targeting multiple fields. Adversaries exploit users' typographical errors to deceive them into accessing or downloading harmful content from the Internet. In this study, we investigate a new type of squatting attack called shift-squatting, wherein adversaries exploit situations in which users inadvertently displace their hands from the intended typing position. This results in multiple characters shifting together in a particular direction on the keyboard. We perform large-scale measurement experiments across three fields: (1) domain names, (2) mobile apps, and (3) software packages. Specifically, we uncovered shift-squatting cases in all three fields, examining over 3 million domain names as well as over 500k mobile apps and Python software packages. Among these, we find that 150,872 shift-squatting domain names have been registered, and some of them are malicious. We also find 6,457 shift-squatting Android apps on the Google Play Store that are marked as potentially harmful for at least one version, and 103 packages are shift-squatted by known malicious packages. These findings indicate that attackers might have already leveraged shift-squatting to distribute malicious contents, and it would become a prevalent security threat if insufficient attention is given.**

I. Introduction

In recent decades, extensive research has been conducted on cybersquatting, leading to the identification of various types of squatting attacks, including typosquatting [1]–[5], combo-squatting [6], bit-squatting [7], [8], and homo-squatting [9], [10]. The primary goal of these squatting attacks is to mislead users into accessing and downloading malicious content. Such attacks present substantial cybersecurity threats, including malware distribution, phishing , and unauthorized data access.

Typosquatting exploits common typographical errors made by users. Numerous studies have shown that users are more likely to make typing mistakes where the misspelled domain name differs by only one character from the original. This concept is often referred to as a Damerau-Levenshtein (DL) distance of one, meaning that the error involves a minimal modification, such as a single-character insertion, deletion, substitution, or transposition. Building on the concept of typosquatting with DL-1, this study introduces a novel attack called shift-squatting, which occurs when a user's hands are inadvertently shifted by one position from their intended placement on the keyboard. This hand shift happens frequently when users are tough typing, where users rely on muscle memory and the consistent placement of fingers on designated keys to type without looking at the keyboard. Unfortunately, the slight hand misalignment can result in multiple simultaneous typing errors, as the user may unintentionally press a sequence of adjacent keys incorrectly. Unlike typosquatting, which typically involves single-character mistakes, shift-squatting captures the more complex scenario where a consistent shift in hand position leads to a pattern of errors. These errors can be exploited by attackers to create malicious content that resembles the intended target.

In this paper, we present a systematic study on shift-squatting and its potential security threats. Based on possible hand movements, we generate a total of 48 different shift-squatting scenarios. Our study further investigates the prevalence of shift-squatting across three fields: (1) domain names, (2) mobile apps, and (3) software packages. First, we crawl as much benign content as possible from the Internet, including 214,848 domain names from the Alexa Top 1 Million list, 59,781 mobile apps from the Google Play Store, and all software packages (570,837) from the Python PyPI registry. Next, we generate shift-squatting counterparts for each of the domains, apps, and software packages, including 3,626,779 domain names and 518,323 app names. Finally, we check the existence of these shift-squatted names and analyze if they contain malicious content or vulnerabilities.

Our results indicate that shift-squatting is prevalent across all three fields, and one-hand shifting is more prevalent than shifts involving both hands. Users might easily obtain unintended objects (e.g., mobile apps or packages) with shift-squatting mistakes. Additionally, we find adversaries might have already leveraged shift-squatting to distribute malicious content. Specifically, we generate shift-squatting domain names and discover that 150,872 of them have DNS and Whois records, suggesting that these domains have already been registered. From a random selection of 500 existing shift-squatting domains, 56 (11%) are identified as malicious. Similarly, we found 226,946 Android apps returned by the Google Play Store when using shift-squatted app names as search inputs, of which 6,457 apps contained malicious code in at least one of their versions. We also identified 34,456 shift-squatting pairs in the Python PyPI registry, and 103 packages are shift-squatted by malicious packages. Overall, our paper makes the following contributions:

- We conduct a large-scale measurement study on shift-

squatting, where attackers exploit potential hand misplacement during typing to lure users into accessing or downloading malicious content. Our findings demonstrate that shift-squatting encompasses a broader range of attack scenarios compared to the well-studied typosquatting, as multiple characters can differ due to shifts in either or both hands in various directions.

- We investigate the shift-squatting threat across three different fields: (1) domain names, (2) mobile apps, and (3) software packages. Our findings reveal that shift-squatting is prevalent in all three areas, with numerous shift-squatting domain names, apps, and packages already in existence.
- We uncover that many of the existing shift-squatting domain names, apps, and packages contain malicious content. This presents significant security concerns, as users may unintentionally access or download harmful content due to shift-squatting errors.

II. Background

Domain Name System (DNS). DNS serves as the primary method for users to access online services, playing a critical role in resolving human-readable domain names into machine-readable IP addresses. When a user types a URL into their browser, the DNS process begins by querying various DNS servers until the correct IP address is found, allowing the browser to establish a connection with the server hosting the requested online service. However, the domain resolution process lacks error-checking functionality, meaning that if a user enters an incorrect or unintended domain name, the system may return a different IP address. This makes domain names a popular target for squatting attacks. If a user mistakenly types an incorrect domain name, they may end up visiting an unintended domain controlled by an adversary, which may contain malicious content.

Mobile Apps. Mobile apps are software applications designed specifically for mobile devices, such as smartphones or tablets, and they have become an integral part of daily life for many individuals. Mobile apps are primarily hosted on app stores, with the Google Play Store being the most popular platform globally. To download an app, users need to type the desired app name into the search bar of the app store. Based on this input, the store's system searches its database and presents the corresponding app for the user to review and download. This highlights the importance of correctly spelling app names, as entering an incorrect name may result in downloading an unintended application. Recent reports and studies have revealed squatting attacks on mobile apps, particularly typosquatting attacks [11], [12]. Adversaries upload malicious apps with typosquatting names to various app markets, luring users into downloading these apps.

Software Packages. Packages are often hosted in registries such as PyPI for Python and npm for JavaScript, which provide users with a centralized platform to search for and download the corresponding packages. In 2021, the top four software registry ecosystems have maintained more than 37 million different versions of packages and attracted 2.2 trillion download requests [13]. Package users (e.g., software developers) utilize registry clients to integrate packages into their software. By typing the package names, registry clients resolve and install the target package and its dependencies from registries. Within the registry, a software package can be identified by package name and its version, where the version number can be omitted when downloading a package. Thus, any typing mistake in entering the package name could result in retrieving an incorrect or unintended package, potentially causing issues such as security vulnerabilities or erroneous software behavior.

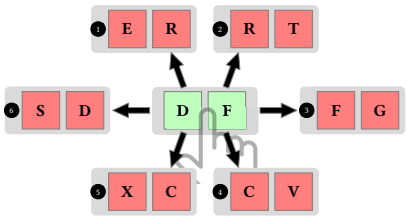

Fig. 1: Illustration of hand shift in six different directions.

III. Overview

Shift-squatting takes advantage of the possibility that a user's hands may become misaligned on the keyboard while typing, and even a slight shift in hand position can lead to numerous unintended errors. When the hands are misaligned but the muscle memory remains intact, the same typing motions may result in incorrect keystrokes. Figure 1 illustrates an example of hand shifting in six different directions, as shown in cases ❶-❻. In the standard English QWERTY layout, each key is surrounded by six adjacent keys, making it easy for a slight misalignment to cause errors. For instance, when the left hand is correctly positioned, the middle and index fingers press the letters `D` and `F`. However, if the hand shifts one position to the top right (as in case ❶), those same fingers will press `E` and `R` instead. In this study, we focus on hand shifts of just one position from the standard placement, consistent with DL-1 that is commonly highlighted in typosquatting research.

Hand shifting can occur in one or both hands. Table I illustrates all possible shift-squatting cases for both left- and right-hand shifts, using the target string *shift*. Cases where only one hand shifts to another position on the keyboard are represented in the first row and first column. Specifically, the first row shows instances where the right hand shifts to one of six positions while the left hand remains in the correct position, and the first column shows instances where the left hand shifts while the right hand remains correctly positioned. The top right corner of the table, where both `L` and `R` are marked with ✗, represents the case where no hand is shifted. Since each hand can shift in one of six possible directions, the table contains a total of 48 shift-squatting scenarios.

A. Threat Model

Typing errors caused by hand shifting can be exploited by attackers to launch shift-squatting attacks. Prior work has studied typing errors and error detection on physical

	R: ✗	R: ❶	R: ❷	R: ❸	R: ❹	R: ❺	R: ❻
L: ✗	shift	sy8ft	su9ft	sjoft	snkft	sbjft	sguft
L: ❶	whir5	wy8r5	wu9r5	wjor5	wnkr5	wbjr5	wgur5
L: ❷	ehit6	ey8t6	eu9t6	ejot6	enkt6	ebjt6	egut6
L: ❸	dhigy	dy8gy	du9gy	djogy	dnkgy	dbjgy	dgugy
L: ❹	xhivg	xy8vg	xu9vg	xjovg	xnkvg	xbjvg	xguvg
L: ❺	zhicf	zy8cf	zu9cf	zjocf	znkcf	zbjcf	zgucf
L: ❻	ahidr	ay8dr	au9dr	ajodr	ankdr	abjdr	agudr

TABLE I: Example of shift-squatting for left and right hand with original string: "*shift*".

keyboards [14], [15], as well as hand drift and error simulation in touchscreen typing [16], [17]. Our work complements these studies by identifying shift-squatted resources flagged as malicious or vulnerable across multiple ecosystems. Specifically, attackers target platforms where users must manually type in the name of a resource to access or download it. In our threat model, attackers are regular Internet users who can register domain names or upload apps and software packages under chosen names through existing platform interfaces. The victims are users who intend to access or download a benign resource by manually typing its name, but their hands may shift by one position on a standard English keyboard layout. The attack goal is to make the shifted input resolve to an attacker-controlled resource, causing the victim to visit, install, or import unintended content. In this study, we focus on investigating shift-squatting on the following three platforms:

- **Domain Names:** A mistyped URL can redirect users to malicious or unintended websites, potentially exposing them to phishing attacks, malware, or fraud.
- **Mobile Apps:** In app stores, a shifted input could lead users to download an entirely different application, which may resemble legitimate apps but contain malicious content.
- **Software Package:** Shift-squatting on software packages can result in importing or installing compromised libraries or tools, jeopardizing software integrity and data security.

To launch a shift-squatting attack, attackers first need to select the platform they wish to target, and then identify the specific resource they want to exploit using shift-squatting. Next, attackers generate squatting names for the chosen target based on different shift-squatting scenarios. Each target can yield a total of 48 potential squatting names, which attackers can register on the chosen platform. Once these squatting names are registered, attackers can host malicious content under these names, luring users who inadvertently mistype due to hand shifting.

Attackers can further enhance the perceived legitimacy of their malicious content by using various techniques to deceive users and bypass security measures. For example, since they control the shift-squatted domain names, they can invest in crafting malicious content that appears authentic and trustworthy. They can also register valid SSL certificates for their shift-squatted domains. This step creates a false sense of security, as many users associate the presence of an SSL certificate with the legitimacy of a website, assuming it has passed authenticity checks.

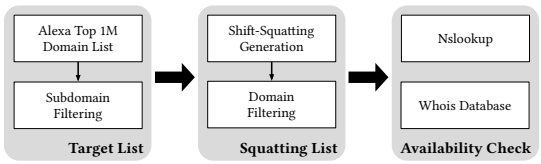

Fig. 2: Overview of experiment design for domain names

B. Ethics

Our experiments are designed to understand and reveal the prevalence of shift-squatting in domain names, mobile apps, and software packages, which do not involve any human subjects nor the collection of any personally identifiable information. Therefore, institutional review board (IRB) approval is waived for our study. Additionally, we refrain from launching any form of proof-of-concept attacks to demonstrate the effectiveness of squatting methods. Publishing this study allows affected stakeholders to become aware of shift-squatting and develop appropriate defenses before it is widely abused.

Our extensive measurement study may inherently generate substantial network traffic, which can potentially affect users within the network. To address this issue, we incorporate time delays in our programs to prevent overloading the queried resources. Furthermore, all experiments are conducted within the university campus, with prior notification and approval from the IT department. We schedule our experiments during off-peak periods to minimize the likelihood of network interference of regular university and student activities.

IV. Shift-Squatting on Domain Names

A. Data Collection

Figure 2 illustrates an overview of our data collection process to investigate the potential threat of shift-squatting in domain names. Specifically, we employ a three-step methodology. First, we identify existing domains that can be targeted in shift-squatting attacks, referred to as the target list. Using this target list, we then generate a squatting list, which contains all possible squatted domain names derived from the target list. Finally, we check the availability of the squatted domains to uncover the overall prevalence of shift-squatting domain names in the wild.

Target List. To construct our target domain list, we first obtain the Alexa Top 1 Million domain list [18]. This dataset offers a wide range of popular domains, making it ideal for analyzing the shift-squatting threat on frequently visited websites. Specifically, we focus on domains consisting solely of second-level domains, excluding any domains with subdomains. This is because adversaries may not have the ability to compromise second-level domains to add malicious subdomains. Instead, they can easily create and register shift-squatted second-level domains directly through domain registrars. After filtering out subdomains, we identify 214,848 second-level domains, which we use as our target list for investigating domain name shift-squatting.

Squatting List. Based on the 214,848 domains in the target list, we apply the 48 shift-squatting scenarios discussed in Sec-

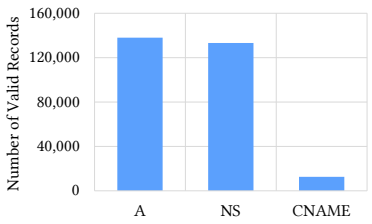

Fig. 3: nslookup results with the number of A, NS, and CNAME records.

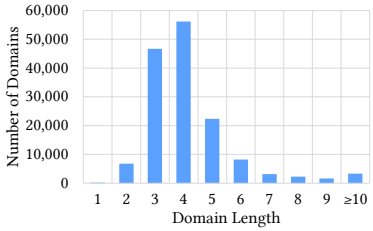

Fig. 4: Distribution of the number of domains in various character lengths.

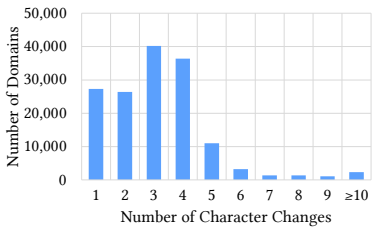

Fig. 5: Number of domains with various character difference.

tion III to generate potential squatted domain names. However, some of the generated squatted domains include keystrokes that are not valid for domain names, such as function keys, CapsLock, Control, Alt, and Win keys. To address this, we apply a domain filtering process to eliminate these invalid keystrokes. After filtering, we obtain 3,626,779 valid shift-squatting domains, which we consider our final squatting list for further analysis.

Availability Check. We check the availability of the domains in the squatting list by querying two different platforms. First, nslookup provides a simple method to check the DNS records of a domain, allowing us to determine whether a domain has been registered. In this study, we focus on the existence of three specific DNS records: (1) **A Record** is used to resolve a domain name to an IPv4 address, (2) **NS Record** identifies the authoritative name servers for the domain, and (3) **CNAME Record** is used to alias one domain name to another domain. If either of these records exists in the nslookup, we consider the domain registered. We also query the Whois database [19] to further verify whether the domains have been registered by checking the domain registrar's information. In addition, the Whois database provides other valuable information for our study. For instance, the Whois database can reveal whether a domain is parked by its owner.

B. Result and Analysis

Our availability check reveals that 150,872 shift-squatted domains have valid DNS records. Figure 3 illustrates the number of A, NS, and CNAME records retrieved. Specifically, 112,359 (74%) domains contain both A and NS records, while only 12,667 (8%) domains include CNAME records. This suggests that the majority of the identified shift-squatting domains are used to host content or services, rather than functioning as placeholders for redirection to other domains.

Our Whois record check reveals that 5,365 domains with valid DNS records are parked domains with parking pages containing various advertisements. 2,403 domains have parking pages that allow users to bid on the domain names at premium prices ranging from $50 to over $2,000. Additionally, we discovered 73,944 shift-squatting domain names that have been registered, but they do not have any valid DNS records. While the purposes of these registered domains remain unclear, we expect that some may be intentionally registered to defend against shift-squatting attacks.

Figure 4 shows the distribution of domain name lengths for all existing shift-squatting domains. In this analysis, only the characters before the top-level domain (TLD) are counted. Among all existing shift-squatted domains, shorter domain names with 5 characters or less account for 132,223 domains, or 88% of the total, suggesting that shorter domain names are more likely to have shift-squatting counterparts. However, we observe that the majority of these domains consist of 3 to 5 characters, with very few domains only one or two characters long. This contradicts our expectation, as shorter domain names would seem more susceptible to shift-squatting. We attribute this phenomenon to two factors: (1) the total number of domains with one or two characters is small, and (2) such domains are often highly popular and come with a large premium for purchase and hosting.

We also analyze the number of character differences between the target domain names and their shift-squatted counterparts. Specifically, 141,375, or 94%, of the shift-squatting domains have five or fewer character differences compared to the target domain names. Notably, squatting domains with only one character difference account for 27,298 or 18% of the total identified shift-squatting domains. This suggests that shift-squatting covers a broader range of scenarios with multiple character changes at once due to hand shifts. In addition, a majority of shift-squatting domains contain multiple character changes, which is different from the well-studied typosquatting, especially bitsquatting attacks focusing on only one character change.

Moreover, Figure 6 illustrates the distribution of shift-squatting domains across different TLDs. Domains with the `.com` TLD dominate the distribution, with 80,992 domains, accounting for 54% of all identified shift-squatting domains. Other commonly observed TLDs include `.org`, `.net`, and various country-level TLDs, but the number of squatting domains is significantly lower compared to `.com`.

Furthermore, Figure 7 shows the various shift patterns and the number of identified domains in each category. Among the 203,755 occurrences, the most common shift pattern is the right hand shifting left while the left hand remains stationary, accounting for 11,131 or 5% of the cases. Six of the top 10 shift-squatting scenarios involve a shift in only one hand, either the left or right, while typing. This suggests that only one hand shifts in various directions remain the primary patterns in current shift-squatting domains.

We also investigate whether the identified shift-squatting

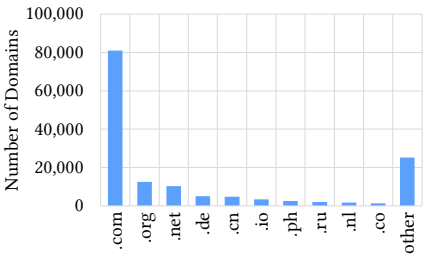

Fig. 6: Top TLD distributions for shift-squatted domain names.

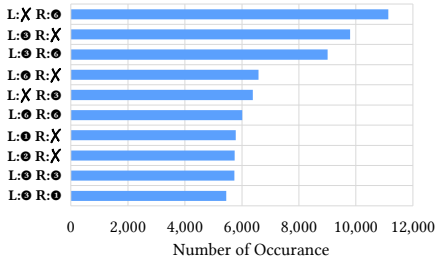

Fig. 7: Top 10 shift-squatting patterns in domain names. (✗ and ❶-❻ in Figure 1)

domains contain malicious content. To do this, we randomly selected 500 domain names and queried VirusTotal [20] for potential malicious activity. As a result, 56 out of 500 domains, or 11%, are detected as malicious. This indicates that shift-squatting could have already been exploited by attackers, and many of these shift-squatting domains are actively being used to host malicious content. Additionally, we also find that even though VirusTotal flags these shift-squatting domain names as malicious, victims can still access these domains without receiving any browser warnings. This indicates that many modern browsers do not have strong enough protective features to defend users against domain shift-squatting.

V. Shift-Squatting on Mobile Apps

A. Data Collection

Similar to the methodology outlined in Section IV-A, we investigate shift-squatting in mobile apps through a three-step process: constructing a target list, generating a squatting list, and checking squatted apps for vulnerabilities. Figure 8 provides an overview of our methodology for studying the prevalence of shift-squatting in mobile apps. In this study, we focus specifically on the Android platform and investigate apps that are publicly available on the Google Play Store [21].

Target List. To construct our target app list, we first download an English dictionary [22] containing a total of 370,099 words. The Google Play Store enforces a rate limit on queries to prevent database overload, making it infeasible to search for every keyword in the dictionary. To address this, we randomly select 74,020 (20%) words from the dictionary to serve as search keywords. We utilize Google Play Scraper [23] to query the Google Play Store and record all apps returned by the search results. This allows us to systematically collect data on the apps available in the store based on the selected keywords. As a result, we obtain 59,781 legitimate apps from the Google Play Store, which we use as our target apps for further analysis.

Squatting List. We apply the 48 shift-squatting scenarios to every app in the target list to generate squatting app names. We then filter the generated names to remove any invalid keystrokes, such as function keys, CapsLock, *etc.*. Due to the same rate limit mentioned before, we randomly select 20% of the filtered squatting app names, resulting in a total of 518,323 app names. We then use Google Play Scraper to query these

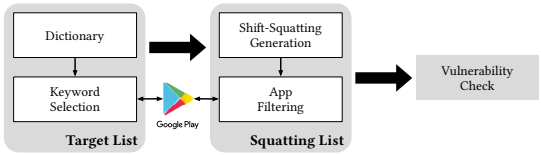

Fig. 8: Overview of experiment design for mobile apps.

squatted app names, and we obtain a total of 5,344,113 apps returned by the scraper. Not that Google Play might return multiple apps within one search. We record both app names and app IDs for these results to create our squatting app list.

Vulnerability Check. Finally, we check whether the apps in our squatting list contain vulnerabilities. To do this, we cross-reference all the app IDs from the squatting list with the AndroZoo database [24], which contains detailed information on 21,817,659 APKs from the Google Play Store. We specifically focus on the `vt_detection` value, which indicates how many times an APK has been flagged as malware by VirusTotal. This information helps us identify malicious apps and determine which Google Play Store apps are vulnerable to shift-squatting attacks.

B. Result

Figure 9 presents the number of squatted apps returned by Google Play for each of the 48 shift-squatting cases. The most frequent case involves the left hand shifting to the right while the right hand remains unshifted, which occurs 5,494 times. Notably, four of the top 10 cases do not involve both hands shifting simultaneously, indicating that one-hand shifting is more prevalent than shifts involving both hands. This pattern is also consistent with the domain name scenario.

We also find that out of the 5,344,113 apps in our squatting list, 9,206 shift-squatting apps are identical to their original target app name in the target list. This indicates that even when users type app names with shifted hands, they can still reach the correct app on the Google Play Store. Figure 11 illustrates the distribution of apps based on the number of character differences between the original app names and their shift-squatted counterparts. The highest proportion of cases occurs when the original and squatted app names differ by two characters, accounting for 3,404 instances. This is followed by 3,008 cases in which the names differ by four characters. The maximum character difference that Google Play is able to correct is 18 characters.

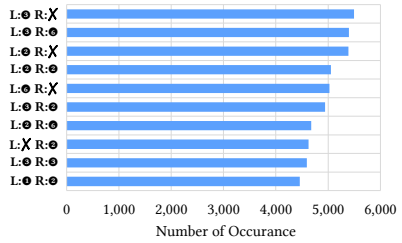

Fig. 9: Top 10 shift-squatting patterns in mobile apps. (✗ and ❶-❻ in Figure 1)

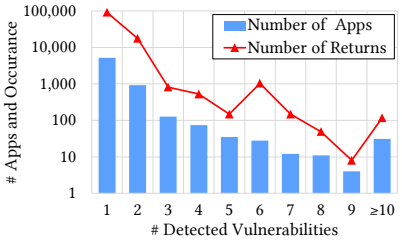

Fig. 10: Number of app occurrence in various detected vulnerability.

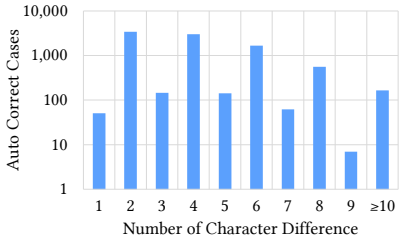

Fig. 11: Number of auto-correct cases with various character differences.

The results of our vulnerability check further reveal that, for 6,457 out of 5,344,113 shift-squatting apps, at least one version is identified as vulnerable by AndroZoo. These vulnerable apps appear a total of 111,849 times in the search results. Figure 10 shows the distribution of apps and the frequency of their appearance based on the number of detected vulnerabilities. In both the line and bar graphs, apps with a detection value of 1 dominate, with 5,211 apps (81%) appearing 91,812 times (82%). 25 apps have a detection value greater than 10, and these appear 76 times.

Furthermore, many of these vulnerable apps have been downloaded frequently. Among the apps with more than 10 detected vulnerabilities, 22 apps (71%) have been downloaded more than 10,000 times, and 5 apps have exceeded 1 million downloads according to the Google Play Store. This highlights the security risk posed by shift-squatting attacks, as users may unintentionally download harmful apps, exposing them to significant threats.

C. Analysis

The results of our experiment show that shift-squatting is a more prevalent and significant attack vector in mobile apps compared to the traditional typosquatting threat. As illustrated in Figure 9, the Google Play Store returns as many apps as it can find that potentially match the search term. On average, shift-squatted search terms return 10 mobile apps. This suggests that even when users make typographical errors due to hand shifts, they are still presented with multiple app options. Our vulnerability analysis indicates that vulnerable shift-squatted apps hosted on the Google Play Store can be accessed by users when search terms match squatted names. This highlights that while Google Play Store conducts comprehensive vulnerability checks for its hosted apps, it lacks the mechanisms necessary to prevent apps from being distributed through shift-squatting.

Interestingly, our experiment reveals that Google Play Store's app search algorithms have the potential to detect and correct shift-squatting errors, successfully handling typing errors with up to 18 characters of difference. This finding exceeds our expectations, as Google Play can accurately and successfully identify the correct app even when the search string is significantly different from the actual app name. However, we also observe that the error correction rate remains low and the corrections appear to occur inconsistently. As a result, users are still at risk of encountering shift-squatting apps on the platform, despite the system's ability to correct some of these errors.

VI. Shift-Squatting on Software Packages

A. Data Collection and Processing

In this study, we investigate the prevalence of the shift-squatting threat in the official Python package registry (*i.e.*, PyPI [25]). To collect all package names and their associated information from PyPI, we employ the same methodology as described in [26]. In total, we retrieve approximately 570,000 package names using the package index list provided by PyPI. In addition, we collect metadata for all identified packages, including their initial release dates, number of downloads, and other relevant information.

Using the 48 shift-squatting cases discussed in Section III, we generate shift-squatting variations of every package name in the PyPI registry. We filter out cases where the generated package names contain invalid characters or keystrokes, such as function keys and Tab key. We cross-reference the generated package names with our original list to identify shift-squatting pairs. In each identified shift-squatting pair, we consider the package with the earlier release date to be the original, and the package released later is considered to be the potential shift-squatting variant. We believe that the original release date is the most accurate and representative metric because it remains unchanged even in the event of package updates.

Finally, we obtain a list of 7,016 unique malicious packages from three malicious software package databases [27]–[29]. We then compare all the shift-squatting pairs we identified with the malicious package lists. This comparison enables us to assess the prevalence of shift-squatting in software packages and to identify cases where shift-squatting attacks have already been launched to distribute malicious packages.

B. Results and Analysis

Among 570k packages, we identify 34,456 shift-squatting pairs. The red line in Figure 12 presents the CDF on the number of shifted characters of these package pairs. While most of them (28%) only shifted one character, which is similar to normal typosquatting, a significant number of them (72%) have shifted multiple times. We further identify 19,642 cases where multiple fingers of one hand have shifted. These results demonstrate that it is highly probable that users download an unintended package with shift-squatting.

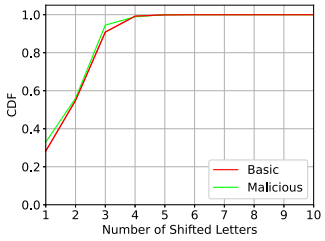

Fig. 12: CDF of shifted characters of packages in PyPI.

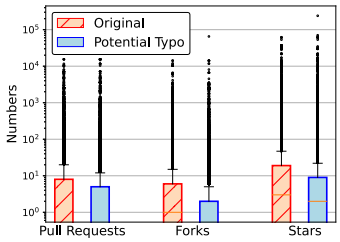

Fig. 13: Statistic data for Shift-Squatting Pairs.

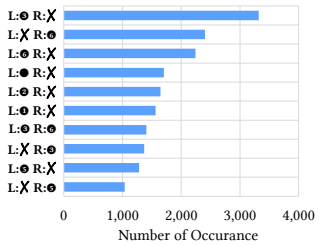

Fig. 14: Top 10 shift-squatting patterns in PyPI. (✗ and ❶-❻ in Figure 1)

The average download of original packages (i.e., package released earlier) is more than 6.4 million, where the number of shift-squatting packages is only about 500k. Figure 13 further illustrates the boxplots of these shift-squatting pairs in terms of total package pull requests, number of forks, and number of stars. All numbers of original data are larger than the shift-squatting packages. The results clearly show that, in general, the original packages are much more popular than the shift-squatting ones.

Finally, Figure 14 shows the frequency of the most popular shift-squatting patterns. The most popular case is left hand shifting right while no shift for the right hand, which occurs 3,316 times. The second popular case is no shift in left hand but right hand shift left (2,402 occurrences). Overall, one hand shifting is more frequent than both hands shifting, and the parallel shifting is more frequent than other shifting cases.

Malicious Package Shift-Squatting. We have identified 103 cases where malicious packages shift-squatted benign packages. We find that the most shift type is the left hand shifting case: there are 11 occurrences of left hand shifting while no right hand shift. The green line in Figure 12 shows the CDF of the number of shifted characters for these shift-squatting packages (i.e., potential victims). It exhibit same patterns as benign packages.

In particular, among them, we find 70 cases that multiple fingers of one hand have shifted. Some interesting malicious-benign package pairs include {`pywz`, `pyex`}, {`gisi`, `bixi`}, and {`ramram`, `tsktsk`}, where two different fingers from the left hand have shifted. These cases might evade a traditional malicious package detector based on typosquatting if it only focuses on one character or one finger change. Another interesting malicious package is `pywe`. This single package can shift-squatting four benign packages, including `pyer`, `py23`, `py34`, and `pyas`. These results indicate that shift-squatting might have already been abused by attackers for distributing malicious packages.

VII. Discussion

Shift-Squatting vs. Typosquatting. The key difference between shift-squatting and typosquatting lies in the nature of user input error attackers exploit. Whereas typosquatting typically involves only a single character difference, such as substitution, addition, deletion, or transposition, shift-squatting arises from hand-position shifts on keyboard, which can simultaneously alter multiple adjacent characters in a single input. Our experimental results demonstrate that such multi-character alterations are not edge cases but instead represent the dominant pattern across all three ecosystems. Specifically, squatting instances with a single-character difference account for less than 30% of the observed cases in each ecosystem.

Mitigation. In contrast to typosquatting, where small edit distances often are sufficient to identify suspicious names, shift-squatting attacks introduce a larger number of character differences that can evade traditional detection mechanisms optimized for single-character variations. As a result, defenses relying solely on simple typo-generation heuristics may fail to capture a significant portion of shift-squatting attacks. Effective mitigation requires coordinated efforts from multiple stakeholders including browsers, end users, domain registrars, app stores, and package registries. Detection approaches should explicitly model keyboard geometry and multi-character errors rather than treating squatting behaviors as independent character-level errors. The generation process used in our study can directly support such efforts by enabling platforms to flag names matching shift-squatting patterns for review, user warnings, or search-ranking adjustments.

Keyboard Layout and Typing Methods. Shift-squatting is significantly influenced by the type of keyboard layout utilized by users, given that disparate layouts can generate diverse squatting outcomes. This research focuses primarily on shift-squatting in standard English keyboard layout, which is commonly found in many off-the-shelf keyboards and represents the most widely used layout worldwide [30]. Alternative keyboard configurations deviate from the standard English layout and present opportunities for exploitation by shift-squatting attackers who target specific regions or linguistic groups. In the future, we plan to broaden the scope of analysis to encompass a wider range of keyboard layouts beyond the standard English layout, including AZERTY, DVORAK, ergonomic keyboards, and non-Latin input methods, which can produce different neighboring-key relationships.

Shift-squatting may also occur on touchscreen devices, where limited screen size and asymmetric thumb reach introduce different systematic error patterns. The mobile shift-squatting is also not limited to on-screen keyboards, as users may search app stores through web interfaces, tablets, or devices paired with physical keyboards. A comprehensive analysis of mobile shift-squatting behaviors would require

modeling of user interaction patterns and likely require human-subject studies, and we plan to explore these in the future.

Proof-of-Concept Shift-Squatting Attack. Our study offers a comprehensive measurement of the prevalence of shift-squatting across various areas, covering all 48 possible squatting scenarios on a standard English keyboard layout. We conservatively treat matched domains, apps, and packages as potential shift-squatting cases, since some overlaps may arise from abbreviations, multilingual conventions, or project-specific choices rather than adversarial intent. One such case is a shifted string that is itself a meaningful word. To quantify this, we apply the same shift-generation process to the English dictionary [22]. Only 0.13% of generated strings coincide with another valid word, indicating that such collisions are rare. Even these cases remain a viable attack surface, as a user with misaligned hands is still directed to an unintended resource. We therefore retain them in our measurement results. Our measurement approach enables a thorough and ethically responsible investigation of shift-squatting while avoiding ethical and practical concerns. Proof-of-concept attacks, although informative, can be intrusive and may inadvertently expose a large population of users to harm. By avoiding launching proof-of-concept attacks, our study minimizes potential risks to end users and infrastructure, while providing insights into the prevalence and structure of shift-squatting attacks.

VIII. Related Work

Domain Name Security. Over the past decade, abusing domain names through typosquatting has been studied from many perspectives [1]–[5], including bitsquatting [7], [8], homograph-based squatting [9], homophone-based squatting [10], abbreviation squatting [31], and combosquatting [6]. These studies demonstrate how adversaries exploit predictable human errors, such as mistyping, misreading, or misinterpreting, to launch large-scale domain squatting attacks. Beyond traditional domains, prior work has also examined typosquatting in International Domain Names [32], non-English keyboard layouts [33], blockchain naming systems [34], and blockchain-based decentralized application scams [35]. Our work complements and extends existing research by identifying and characterizing shift-squatting as a previously unexplored squatting vector. To the best of our knowledge, this work provides the first comprehensive measurement and analysis of shift-squatting as a distinct threat model.

Mobile App Security. Mobile apps have been demonstrated to be vulnerable to typosquatting. Hu *et al.* [12] studied both mobile app name squatting and package name squatting, which motivates our work to investigate shift-squatting in mobile apps. Extensive research efforts have also addressed mobile app' security and privacy issues [36]–[39]. In particular, Android apps are known to be easily repackaged [40], where attackers decompile a benign app, insert malicious payload, and further package the app into mobile markets. Additionally, attackers create fake apps where malicious apps look similar to legitimate ones [41], [42]. Tang *et al.* [43] collected more than 150K fake apps that have the same package/app names as popular apps. Our work extends these prior studies on mobile app squatting by introducing shift-squatting as a distinct and previously unexplored threat in app marketplaces.

Software Repository Security. Typosquatting [44] is one of the most exploited threats in the software registry ecosystems , along with other threats such as package vulnerability [45], [46], account takeover [47], [48] and infrastructure compromise [49]. Many empirical studies have been conducted on different software ecosystems [50]–[53]. Vu *et al.* [54] systematically studied typosquatting and combosquatting attacks in Python, demonstrating how malicious packages exploit name similarity to deceive users. Zimmermann *et al.* [55] analyzed the security threats in the npm ecosystem, revealing ecosystem-level risks arising from dependency structures and package popularity concentration. Duan *et al.* [56] detected hundreds of malicious packages and found that typosquatting and account hijacking are the two most exploited vectors in PyPI, npm, and RubyGems. Gu *et al.* [26] uncovered that 96.9% of malicious packages that are officially marked by npm are similar to existing benign ones, indicating that typosquatting is already heavily abused by attackers. Our work extends these prior studies and further demonstrates that shift-squatting could also be a threat in existing package registries. Existing defenses mechanisms tailored to traditional typosquatting in package registries is not effective against shift-squatting and may leave a non-trivial attack surface unaddressed.

IX. Conclusion

This paper investigates shift-squatting, a squatting vector in which one-position hand misalignment simultaneously alters multiple characters, producing names that differ substantially from their targets. We have performed large-scale measurement experiments across domain names, mobile apps, and software packages. Our results indicate that many shift-squatting pairs have already existed in the wild, suggesting that shift-squatting is not merely a theoretical concern but an active and emerging attack vector. Users who fall victim to shift-squatting mistakes may be redirected to unintended objects, including malware, phishing content, and fraudulent applications or packages. Our work highlights the need for the security community and ecosystem operators to explicitly account for shift-squatting in future detection, mitigation, and user-protection mechanisms.

Acknowledgements

We thank the anonymous reviewers for their insightful comments, which help to improve the quality of this paper. This work is partially supported by the National Science Foundation (NSF) grants CNS-2317830 and CNS-2338837, the Internet Freedom Fund from the Open Technology Fund (OTF), and the grant from the Coastal node of Virginia's The Commonwealth Cyber Initiative (CCI).

REFERENCES

[1] R. Tahir, A. Raza, F. Ahmad, J. Kazi, F. Zaffar, C. Kanich, and M. Caesar, "It's All in the Name: Why Some URLs are More Vulnerable to Typosquatting," in *IEEE INFOCOM*, 2018.

[2] M. T. Khan, X. Huo, Z. Li, and C. Kanich, "Every Second Counts: Quantifying the Negative Externalities of Cybercrime via Typosquatting," in *IEEE S&P*, 2015.

[3] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich, "The Long "Taile" of Typosquatting Domain Names," in *USENIX Security Symposium*, 2014.

[4] P. Agten, W. Joosen, F. Piessens, and N. Nikiforakis, "Seven Months' Worth of Mistakes: A Longitudinal Study of Typosquatting Abuse," in *NDSS*, 2015.

[5] K. Tian, S. T. Jan, H. Hu, D. Yao, and G. Wang, "Needle in a Haystack: Tracking Down Elite Phishing Domains in the Wild," in *ACM IMC*, 2018.

[6] P. Kintis, N. Miramirkhani, C. Lever, Y. Chen, R. Romero-Gómez, N. Pitropakis, N. Nikiforakis, and M. Antonakakis, "Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse," in *ACM CCS*, 2017.

[7] N. Nikiforakis, S. Van Acker, W. Meert, L. Desmet, F. Piessens, and W. Joosen, "Bitsquatting: Exploiting Bit-flips for Fun, or Profit?" in *WWW*, 2013.

[8] T. Vissers, T. Barron, T. Van Goethem, W. Joosen, and N. Nikiforakis, "The Wolf of Name Street: Hijacking Domains Through their Nameservers," in *ACM CCS*, 2017.

[9] T. Holgers, D. Watson, and S. Gribble, "Cutting through the Confusion: A Measurement Study of Homograph Attacks," in *USENIX ATC*, 2006.

[10] N. Nikiforakis, M. Balduzzi, L. Desmet, F. Piessens, and W. Joosen, "Soundsquatting: Uncovering the Use of Homophones in Domain Squatting," in *International Conference on Information Security*, 2014.

[11] "Fake WhatsApp app fooled million Android users on Google Play: Did you fall for it?" https://www.zdnet.com/article/fake-whatsapp-app-fooled-million-android-users-on-google-play-did-you-fall-for-it/, 2017.

[12] Y. Hu, H. Wang, R. He, L. Li, G. Tyson, I. Castro, Y. Guo, L. Wu, and G. Xu, "Mobile App Squatting," in *The Web Conference*, 2020.

[13] "Sonatype's 2021 State of the Software Supply Chain," https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021.

[14] M. Rieger and V. K. E. Bart, "Typing Style and the Use of Different Sources of Information during Typing: An Investigation Using Self-Reports," *Frontiers in Psychology*, 2016.

[15] S. F. Dahm and M. Rieger, "Errors in Imagined and Executed Typing," *Vision*, 2019.

[16] F. C. Y. Li, L. Findlater, and K. N. Truong, "Effects of Hand Drift While Typing on Touchscreens," in *Graphics Interface*, 2013.

[17] D. Shi, Y. Zhu, F. E. Fernandes Junior, S. Zhai, and A. Oulasvirta, "Simulating Errors in Touchscreen Typing," in *ACM CHI Conference on Human Factors in Computing Systems*, 2025.

[18] Amazon, "Alexa Top 1 Million Domains," https://s3.amazonaws.com/alexa-static/top-1m.csv.zip, 2024.

[19] "WhoisXMLAPI," https://whois.whoisxmlapi.com/, 2024.

[20] "VirusTotal," https://www.virustotal.com/gui/home/upload, 2024.

[21] "Google Play Store," https://play.google.com/store/, 2024.

[22] "List Of English Words," https://github.com/dwyl/english-words?tab=readme-ov-file, 2024.

[23] "Google Play Scraper," https://pypi.org/project/google-play-scraper/.

[24] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting Millions of Android Apps for the Research Community," in *International Conference on Mining Software Repositories*, 2016.

[25] [Online]. Available: https://pypi.org/

[26] Y. Gu, L. Ying, Y. Pu, X. Hu, H. Chai, R. Wang, X. Gao, and H. Duan, "Investigating package related security threats in software registries," in *IEEE S&P*, 2023.

[27] M. Ohm, H. Plate, A. Sykosch, and M. Meier, "Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2020.

[28] [Online]. Available: https://github.com/lxyeternal/pypi_malregistry

[29] "Malicious Software Packages Dataset," https://github.com/datadog/malicious-software-packages-dataset.

[30] [Online]. Available: https://keyshorts.com/blogs/blog/us-keyboard-layout-everything-you-need-to-know

[31] P. Lv, J. Ya, T. Liu, J. Shi, B. Fang, and Z. Gu, "You Have More Abbreviations Than You Know: A Study of AbbrevSquatting Abuse," in *ICCS*, 2018.

[32] B. Liu, C. Lu, Z. Li, Y. Liu, H.-X. Duan, S. Hao, and Z. Zhang, "A Reexamination of Internationalized Domain Names: The Good, the Bad and the Ugly," in *IEEE/IFIP DSN*, 2018.

[33] V. Le Pochat, T. Van Goethem, and W. Joosen, "A Smörgåsbord of Typos: Exploring International Keyboard Layout Typosquatting," in *IEEE S&P Workshops (SPW)*, 2019.

[34] M. Muzammil, Z. Wu, L. Harisha, B. Kondracki, and N. Nikiforakis, "Typosquatting 3.0: Characterizing Squatting in Blockchain Naming Systems," in *Symposium on Electronic Crime Research (eCrime)*, 2024.

[35] P. Xia, H. Wang, B. Gao, W. Su, Z. Yu, X. Luo, C. Zhang, X. Xiao, and G. Xu, "Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange," *ACM Measurement and Analysis of Computing Systems*, 2021.

[36] K. Chen, X. Wang, Y. Chen, P. Wang, Y. Lee, X. Wang, B. Ma, A. Wang, Y. Zhang, and W. Zou, "Following devil's footprints: Cross-platform analysis of potentially harmful libraries on android and ios," in *IEEE S&P*, 2016.

[37] L. Xing, X. Pan, R. Wang, K. Yuan, and X. Wang, "Upgrading your android, elevating my malware: Privilege escalation through mobile os updating," in *IEEE S&P*, 2014.

[38] Y. Nan, M. Yang, Z. Yang, S. Zhou, G. Gu, and X. Wang, "Uipicker:user-input privacy identification in mobile applications," in *USENIX Security Symposium*, 2015.

[39] X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, "Identity, location, disease and more: Inferring your secrets from android public resources," in *ACM CCS*, 2013.

[40] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *ACM conference on Data and Application Security and Privacy*, 2012.

[41] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *IMC*, 2018.

[42] S. M. Kywe, Y. Li, R. H. Deng, and J. Hong, "Detecting camouflaged applications on mobile application markets," in *ICISC*, 2015.

[43] C. Tang, S. Chen, L. Fan, L. Xu, Y. Liu, Z. Tang, and L. Dou, "A large-scale empirical study on industrial fake apps," in *IEEE/ACM ICSE-SEIP*, 2019.

[44] N. P. Tschacher, "Typosquatting in Programming Language Package Managers," 2016.

[45] J. Wetter and N. Ringland, "Understanding the Impact of Apache Log4j Vulnerability," https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html.

[46] "[CVE-2019-15224] Version 1.6.13 published with malicious backdoor," https://github.com/rest-client/rest-client/issues/713.

[47] D. Gruss, M. Schwarz, M. Wübbeling, S. Guggi, T. Malderle, S. More, and M. Lipp, "Use-after-freemail: Generalizing the Use-after-free Problem and Applying It to Email Services," in *AsiaCCS*, 2018.

[48] P. Doerfler, M. Marincenko, J. Ranieri, Y. Jiang, A. Moscicki, D. McCoy, and K. Thomas, "Evaluating Login Challenges as a Defense Against Account Takeover," in *WWW*, 2019.

[49] "Postmortem for Malicious Packages Published on July 12th, 2018," https://eslint.org/blog/2018/07/postmortem-for-malicious-package-publishes.

[50] J. Kabbedijk and S. Jansen, "Steering Insight: An Exploration of the Ruby Software Ecosystem," in *Springer ICSOB*, 2011.

[51] D. M. German, B. Adams, and A. E. Hassan, "The Evolution of the R Software Ecosystem," in *IEEE CSMR*, 2013.

[52] A. Decan, T. Mens, and E. Constantinou, "On the Impact of Security Vulnerabilities in the npm Package Dependency Network," in *MSR*, 2018.

[53] K. Manikas, "Revisiting Software Ecosystems Research: A Longitudinal Literature Study," *Journal of Systems and Software*, 2016.

[54] D.-L. Vu, I. Pashchenko, F. Massacci, H. Plate, and A. Sabetta, "Typosquatting and Combosquatting Attacks on the Python Ecosystem," in *IEEE EuroS&PW*, 2020.

[55] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small World with High Risks: A Study of Security Threats in the npm Ecosystem," in *USENIX Security*, 2019.

[56] R. Duan, O. Alrawi, R. P. Kasturi, R. Elder, B. Saltaformaggio, and W. Lee, "Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages," in *NDSS*, 2021.