Social Network Analysis and Mining (2024) 14:177
https://doi.org/10.1007/s13278-024-01321-x

ORIGINAL ARTICLE

Disambiguating usernames across platforms: the GeekMAN approach

Md Rayhanul Masud[1] · **Ben Treves**[1] · **Michalis Faloutsos**[1]

Received: 29 February 2024 / Revised: 26 July 2024 / Accepted: 26 July 2024 / Published online: 31 August 2024

Abstract
How can we identify malicious hackers participating in different online platforms using their usernames only? Establishing the identity of a user across online platforms (e.g. security forums, GitHub, YouTube) is an essential capability for tracing malicious hackers. Although a hacker could pick arbitrary names, they often use the same or similar usernames as this helps them establish an online "brand". We propose GeekMAN, a systematic human-inspired approach to identify similar usernames across online platforms focusing on technogeek platforms. The key novelty consists of the development and integration of three capabilities: (a) decomposing usernames into meaningful chunks, (b) de-obfuscating technical and slang conventions, and (c) considering all the different outcomes of the two previous functions exhaustively when calculating the similarity. We conduct a study using 1.8M usernames from three different types of forums: (a) security forums, (b) malware authors from GitHub, and (c) mainstream social media platforms, which we use as reference. First, our method outperforms previous methods with a Precision of 81–86% on technogeek datasets. Second, we find 6327 forum users that match malware authors on GitHub with a high similarity score (≥ 0.7). Finally, we provide a translation dictionary for slang terms with 5.8K entries, and create GeekMAN platform to facilitate further studies https://geekman.streamlit.app.

Keywords Username matching · Hacking · GitHub · Cybersecurity · Online forum analysis · Social network

1 Introduction

How can we identify malicious hackers across different platforms? This is the question that motivates our work. First, hackers with visible online personas often lead major cyber-criminal activities (Samtani and Chen 2016). Second, these hackers are active and visible on many online platforms including specialized security forums and popular platforms like GitHub (Islam et al. 2021a). In fact, some of these platforms harbor malicious activities to the point that they are forced to shut down (Gharibshah et al. 2018). One thing is clear: these hackers create a brand around their online names. As a result, hackers: (a) adopt unusual names, and (b) use them fairly consistently with only minor changes across different platforms. For example, a username of interest could be *w33dgod*, which we may want to match with *godweed* (both are real usernames).

The problem we address here is the following: given two usernames, how can we determine if they are likely to belong to the same user? As our focus is tracing hacker activity, we focus on **technogeek** usernames, which we define as usernames with: (a) technical jargon, (b) slang and unconventional use of letters and characters, and (c) multiple parts. These types of usernames seem to be used by malicious hackers, but also by tech-enthusiasts, gamers etc. For example, *IAmBlackHacker* on GitHub and *B14CKH4K3R* on Facebook.[1] We refer to this kind of *obfuscation* using letters and digits in unusual ways as **slangification**. Many of their usernames have multiple parts, which we refer to as **chunks**. Traditional string matching and edit distance techniques have difficulty matching these types of usernames. Here, we impose an additional challenge: we do not use other types of information, such as demographic attributes, context, or social connections, which could help refine the

✉ Md Rayhanul Masud
mmasu012@ucr.edu

Ben Treves
btrev003@ucr.edu

Michalis Faloutsos
michalis@cs.ucr.edu

[1] UC Riverside, Riverside, USA

[1] All examples here are real usernames. The GitHub malware author *IAmBlackHacker* refers to https://www.facebook.com/B14CKH4K3R as her blog in the profile information. The Facebook page claims Varanasi, India as its location, so we suspect that *black* must refer to black hat hacking.

matching accuracy. Our framing of the problem can be seen as an *essential building block* of a broader solution. Thus, our goal is to push the username level matching to its limit: how far can we go with just usernames?

There has been relatively little work on the problem as we define it here. In particular, we find that most of the previous works: (a) focus on the popular social media usernames, (b) rely on training data, and (c) use string matching without following a human-like interpretation, such as decomposing the username into meaningful chunks. Specifically, many methods treat the username as a string, and use features such as the frequency of bi-grams and tri-grams. Overall, we can group the previous efforts in three large families depending on their primary focus on: (a) concentrating on username similarity (Perito et al. 2011; Zafarani and Liu 2013; Wang et al. 2016; Li et al. 2019), (b) leveraging user profile attributes, such as demographic information (e.g. gender, education, job title) (Vosecky et al. 2009; Goga et al. 2013; Zhang et al. 2014; Mu et al. 2016), and (c) combining multiple dimensions of information, such as the user-generated content, including topics and linguistic style, and in-platform social connections (Liu et al. 2013, 2016; Zhang et al. 2018). As we explain later, we compare our approach against a set of state-of-the-art username similarity algorithms (Wang et al. 2016; Li et al. 2019). We discuss previous works in Sect. 7.

As our key contribution, we propose GeekMAN, a systematic approach for linking technogeek users across platforms. Our approach is inspired by human cognition: it attempts to emulate how a human will try to disambiguate this type of username, such as *IAmBlackHacker* and *B14CK-H4K3R*, which we mentioned above. The key novelty of our work consists of the development and integration of three capabilities: (a) **deslangification**, which de-obfuscates slang and *geeky* naming conventions, (b) **chunkification**, which decomposes usernames into meaningful chunks, which leads to one or more lists of chunks, and (c) **comparison**, which considers all the lists of chunks to calculate the similarity between two given usernames. We introduce our metric of Similarity Score, *SimScore* , and we use the Similarity Score Threshold, *SimT*, to select username pairs that are a likely match.

We deploy our approach on 1.8M usernames from three different types of online platforms: (a) five popular hacker-rich security forum users, (b) 7.3K malware authors from GitHub, and (c) three mainstream social media platform users, which we use as reference. The key results are summarized below.

a. Technogeek usernames use slang and chunks extensively. To quantify the use of slang, we compare usernames that have multiple digits or symbols in between characters. We find that technogeek forums have 1.5-6 times more such usernames compared to social media platforms (see Sect. 3). Quantifying the prevalence of chunks, we find that 60-70% of the usernames could be decomposed into at least four chunks, while roughly 20% of them have more than 6 chunks!

b. GeekMAN outperforms prior approaches esp. on technogeek usernames Focusing on technogeek usernames, we find that our approach identifies matches with 86.0% Precision and 72.6% Relative F1-score (which we define later). By contrast, two prior approaches exhibit 76.0% and 47.4% Precision with 42.9% and 57.1% Relative F1-score, respectively. Our approach outperforms prior methods even with usernames of common social networks.

c. GeekMAN finds 6327 forum users who likely own malicious repositories on GitHub! As an indicative use of our approach, we match security forum users to malicious GitHub authors (Rokon et al. 2020). We find 6327 such matches with a similarity score threshold of $SimT = 0.7$. Manual investigation to a set of randomly selected matched users shows that often the forum users proudly point to their own GitHub profile or repositories, which validates the match.

d. GeekMAN allows for balancing the Precision - Recall trade off. Depending on the need of the study, our approach can be easily tuned to favor higher Precision or higher Recall. The 6327 likely matches between forum and GitHub users for $SimT = 0.7$ become 1958 matches for Similarity Score Threshold $SimT = 0.9$. Note that only 260 matches correspond to identical strings: our method adds value beyond the obvious matches. We find that 57.7% of these 1958 matches use digits/symbols in their usernames as opposed to only 16.2% of among the 260 exact matches.

e. An online platform, datasets, and a "deslangification dictionary". To facilitate further research, we provide: (a) an online tool, as shown in Fig. 1, (b) our datasets, (c) our groundtruth, and (d) a slang-translation dictionary. We are off to a great start: with 1.8M usernames, three different

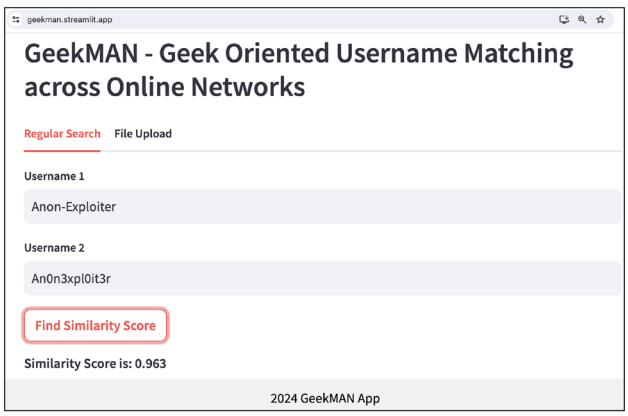


Fig. 1 GeekMAN Platform: Matching usernames *Anon-Exploiter* and *An0n3xpl0it3r*

username-matching algorithms, and manually verified ground-truth. Currently, our slang-translation dictionary has 5.8K entries consisting of slang words and their counterparts. We have shared our source code in GitHub (GeekMAN 2023).

Our work in perspective. Our approach is a fundamental building block for tracing users across different platforms with an emphasis on technogeek usernames. We revisit its practical impact and limitations in Sect. 6. In the same section, we also discuss the issue that a matching pair of usernames does not necessarily mean that they belong to the same user.

2 Background and data

In this section, we provide some background, explain the motivation of our approach, and discuss the dataset in detail.

2.1 Malicious hackers use technogeek in usernames

Hackers and other malicious users often participate in various public platforms, including specialized discussion forums, technical forums, and software platforms like GitHub. Their main goals seem to be: (a) establishing an online brand, and (b) boasting of their accomplishments (Islam et al. 2021a, b). For example, we have seen posts by hackers to advertise their services, claim the credit of successful hacking activities, and create communities of like minded people (Islam et al. 2022).

Often they use the same username consistently across platforms, but sometimes they vary the spelling or structure of the name or the order of the parts of their names. They sometimes choose different username spelling via transformation of their original alias in order to establish a branding or hide their own persona in the forum. We refer to this username spelling transformation as **slangification**. As a noteworthy example, an FBI most wanted cybercriminal, having alias *ha0r3n*, was found to have a GitHub profile named *wo4haoren*. In 2020, FBI listed another most wanted hacker named *Behzad Mohammadzadeh*, with alias *Mrb3hz4d*, who was charged for defacing a number of websites (FBI 2020).

2.2 Datasets

Our dataset contains nine different online platforms spanning across three broad categories: security forums, open-source software plaforms, and social networks. Table 1 displays a summary of them.

Table 1 Summary of datasets: forums, GitHub, and social media

Category	Platform	Abbr.	Users
Security forums	Garage4Hackers	GH	865
	Offensive Community	OC	11371
	RaidForums	RF	44107
	Multiplayer Game Hacking	MP	507945
	Hack Forums	HF	659672
Software platforms	GitHub	GT	7389
Social media	Facebook	FB	163037
	Twitter	TW	196534
	Google+	GP	280318

2.2.1 Security forums

This category comprises five security forums: Garage4Hackers(GH), Offensive Community(OC), RaidForums(RF), Multiplayer Game Hacking(MP), and Hackforums(HF) (Garage4Hackers 2021; Community 2021; RaidForums 2021; Hacking 2021; Hackforums 2021). The data of this category contains posts and threads of 1.2M users ranging between 2005 and 2021. The data comes from two main sources, our automated crawler and Cambridge Cybercrime Centre (cambridgecybercrime 2022), who kindly shared their data with us.

2.2.2 The GitHub software platform

Interestingly, hackers share malware source code in public platforms, such as GitHub. We consider 7389 GitHub (GT) authors, who were identified to have at least one repository with malware source code (Rokon et al. 2020; SourceFinder 2022).

2.2.3 Social media

We also consider 639K usernames (Goga et al. 2015) from three popular social networks: Facebook(FB), Twitter(TW), and Google+(GP) (the popularity of Google+ did not last). Among them there are 49K pairs of usernames that belong to the same user, which we use later in our evaluation. We use this dataset to compare the username patterns in *typical* social networks against the patterns in our technogeek platforms, and to evaluate the performance of GeekMAN in regular usernames. For the rest of the paper, we use the term technogeek to refer to the first two categories: security forums and GitHub.

2.3 Validation and groundtruth

Note that the social media dataset consists of verified accounts owned by the same users. So we use this to partly evaluate our algorithms. However, given our focus on technogeek forums, we describe our validation approach in Sect. 5.

3 Quantifying technogeek usernames

In this section, we study the usernames in technogeek forums, which are likely to be visited by hackers. The goal is to understand the username selection patterns and conventions in order to inform our approach. Overall, we find that the usernames in technogeek forums are different from those in general purpose platforms, such as social media.

3.1 Technogeek forum users use symbols and digits more frequently than social media users

We measure the number of usernames that contain symbols and digits to test our intuition that technogeek usernames contain these characters more frequently and in more unusual ways compared to social media usernames. We plot the distribution of the users in different platforms who use digits in between two alphabetical characters in their usernames in Fig. 2. We see that around 17~37% of technogeek forum usernames have multiple digits in between letters of their usernames, which is at least 1.5-6 times compared to social media users. This indicates that the technogeek users are more likely to use this type of usernames. For example: *n1nj4sec* and *z3r0d4y* contain two and three digits in between letters, instead of *ninjasec* and *zeroday*. Furthermore, we find that 22% of the GitHub users in our dataset, indicated as GT in the plot, also demonstrate this type of behavior. Recall that our GitHub users have been identified as malware producing authors, so this percentage will most likely differ if we consider GitHub users at large. By contrast, this behavior is much less pronounced in social media platforms, such as Facebook. We find similar trends for using symbols during the investigation.

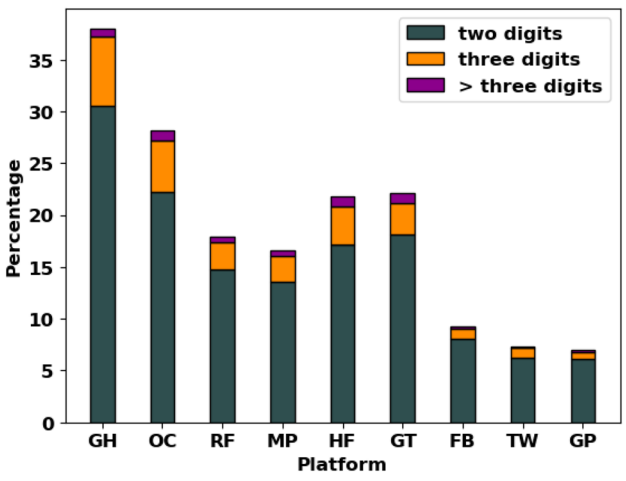


Fig. 2 Distribution of percentage of population who use digit in between letters multiple times. An example username containing two digits in between letters: *g3ntl3man*

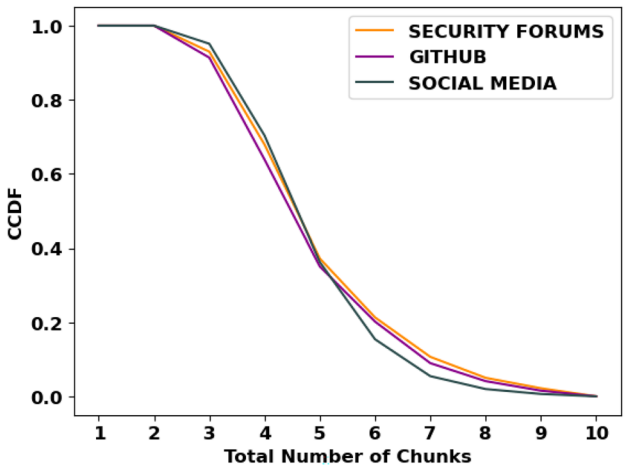


Fig. 3 The CCDF of total number of chunks found per username across different types of online platforms. An example username: *sniper7kills* having three chunks *{sniper, 7, kills}*

3.2 We find 60-70% of usernames with multiple chunks

We want to quantify how frequently usernames consist of multiple parts, which we refer to as **chunks**. We use the Chunkification method that we describe in the next section on all the online platforms in our dataset. For each username, we consider a list of all possible chunks, which we could find from different chunkification approaches. In Fig. 3, we plot the CCDF (Complementary Cumulative Distribution Function) curve of the total number of chunks found per username. We find that approximately 60-70% of the users on each of the online platforms contain 4 or more chunks in their usernames, irrespective of technogeek and social media platforms. We also notice that the number of chunks per username seems consistent across all platforms, although the frequency of more than 5 chunks is slightly higher among the technogeek forums.

In an effort to get a deeper understanding, we also investigated the username chunks. We find that general users prefer proper English words and names in their chunks. By

contrast, technogeek forum users transform the spelling of English words in their usernames by replacing one or more letters with digits/symbols. Our analysis concludes that the percentage of technogeek users who use slangification is 2–8 times higher compared to social media users. We illustrate the most commonly slangified English words used by technogeek forum users via a word cloud shown in Fig. 4.

4 Proposed method

The goal of our method is to determine the similarity score of two given usernames. The inspiration behind GeekMAN is the emulation of a human interpreter. The key idea is to de-obfuscate the usernames (if possible), decompose them into several possible lists of chunks, and finally compare the lists of chunks to calculate the similarity score.

Our approach consists of three main modules: (a) Deslangification, (b) Chunkification, and (c) Comparison. We provide a conceptual overview in Fig. 5 and demonstrate its operation for usernames *z3r0c00l* and *COOL_zERO* in Fig. 6.

4.1 Deslangification module

This module aims to deslangify a given username by: (a) identifying likely use of slang, and (b) translating the slang into regular words. One such example is: *z3r0c00l* which most likely refers to *zerocool*.

4.2 Chunkification module

This module takes a username and produces one or more lists of *chunk*s. We use the term **L** to refer to each such list and the term **Bag** to refer to all the lists of chunks for a username. Table 2 provides a list of our terminology. Note that in its most general case, the Chunkification module can provide more than one possible ways to chunkify the username. E.g.

(a) Technogeek forums (b) Social media

Fig. 4 The word-cloud that shows popular slangified words used at least 10 times as usernames or as parts of usernames in technogeek forums (200 words) than social media (6 words only)

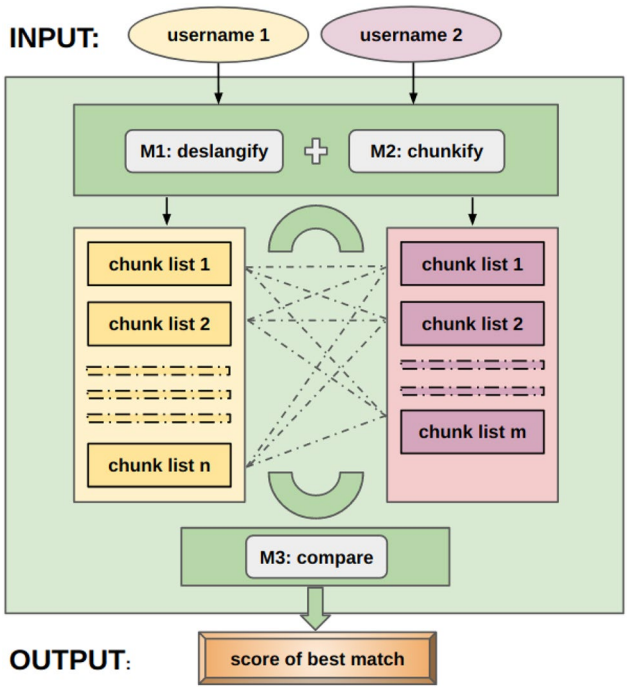


Fig. 5 GeekMAN calculates the similarity score for a pair of usernames focusing on technogeek users. Our approach tries to emulate human interpretation by combining the: **a** Chunkification, **b** Deslangification and **c** Comparison modules

mr-satan1 can be decomposed into: {*mr, -, satan, 1*}, and {*mr, -, satan1*}, since Satani[2] could be a slangified version of someone's last name. In fact, we want to generate as many as possible plausible lists of chunks to ensure the highest possible similarity.

4.3 Comparison module

This module takes as input two Bags with lists of *chunk*s, one for each username, and returns the highest similarity score among all possible pairs of lists. In more detail, it consists of functions that calculate the similarity at different granularity: (a) between two *chunk*s, (b) between two lists of *chunk*s, and (c) between two Bags of lists of *chunk*s.

4.4 Our approach in detail

We now describe our method in more detail by discussing our algorithmic choices and the related challenges. For convenience, we list key terms and functions in Table 2.

[2] According to Wikipedia, Satani is the name of a Vaishnavite community and caste in India, and also a community in Northern Ghana.

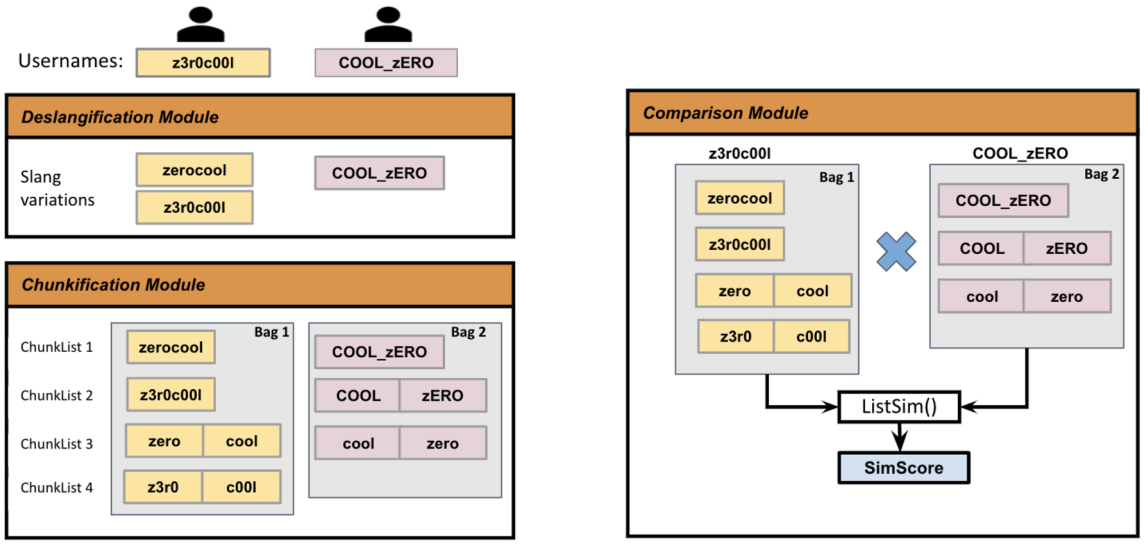


Fig. 6 An example of how the modules of our approach will handle a pair of usernames: *z3r0c00l* and *COOL_zERO*

Table 2 Definition of terminology for GeekMAN approach

Terms	Definitions
alpha	A lowercase or uppercase English letter.
num	A numerical character.
sym	A non-alphanumeric character.
char	A character: *alpha*, *num*, or *sym*.
chunk	A string of *char*s.
L	A list of *chunk*s.
Bag (*u*)	A set of L generated from username *u*.
slangChar	A single *num* or *sym* which is known to be used in place of an *alpha*.
slangCharMap()	A function that maps a potential *slangChar* to *alpha* characters. E.g. "3" to "e" and "3" to "s".
TokenDict	A dictionary of English word/name phrases.
ChunkSim()	A function that returns the similarity between two *chunk*s.
ListSim()	A function that returns the similarity between two L.

4.4.1 Maximizing the likelihood of a match

The task of reverse engineering the naming habits of users is a challenging problem. The overarching goal is to ensure that we find a possible match, even in the face of obfuscation. The two modules of deslangification and chunkification can be used synergistically and in conjunction. To have the broadest possible coverage, we can consider the following approaches:

1. We do deslangification and then chunkification
2. We do chunkification and then deslangification
3. We keep the initial username as is as a one-chunk list
4. We keep the deslangified username as one-chunk list
5. We filter out unlikely lists of chunks (optional)
6. We apply chunkification iteratively on chunks

Note that the best results are derived from the first sequence in our study as we explain in Sect. 5.

4.4.2 Deslangification module

In this module, we try to de-obfuscate the username to identify any available slangified *chunk*. First, we create *slangCharMap*(), a function that maps a potential slang character to a letter following the common technogeek conventions based on our observations and commonly reported usage (Wikipedia 2023). Then, we search for potential slang characters in the username, and if any are found, we replace them with the corresponding letter character found from

slangCharMap(). As an example, username *th3m4lw4r3* can turn into *themalware* and username *z3r0s4mur41* into *zerosamurai*. We notice that digit 4 can be translated as both a and r. In addition, digit 4 does not necessarily have to transform into a letter. Thus, we obtain a plethora of potential deslangified versions from one username along with the original username string.

4.4.3 Chunkification module

We perform chunkification on a given username based on different criteria and create a Bag which is a set of lists of *chunk*s. We consider four different chunkification criteria emulating four different aspects of username naming behavior. The algorithmic challenge is to strike the following balance. On the one hand, we want to extract as many as possible *reasonable* lists of *chunk*s to maximize the possibility of similarity. On the other hand, doing all possible chunkifications without any semantic cues would create a combinatorial explosion of lists. We address this problem with the following methods.

a. Symbol-based chunkification: Some symbols are used as delimiters to chunkify usernames. Typically delimiter symbols are the underscore, the space and the dot. In Section 3, we discussed the idea of using symbols in usernames. We create L, a list of *chunk*s using the symbols present in the username. An example is *COOL_zERO* which we can split into {*COOL, zERO*}. Note that we could of course keep delimiter symbols as chunks in the list, but we consider that they would rather impede than help matching accuracy.

b. Digit-based chunkification: Numbers are sometimes used to separate words, and we use this here as well. Digits in a username can be either something telling about the user or just a combination of single/ multiple random digits. For example, in one case a user may choose to use digits related to her personal information, while in another case a user may use random digits simply to make her username comply with certain platform requirements. We search for numbers in the username, and if any are found, we chunkify the username accordingly. An example of digit based chunkification can be: *sniper7kills* will be split into {*sniper, 7, kills*}.

c. Capitalization-based chunkification: Capital letters can also provide cues for chunks. Capitalization is often used by users to specify multiple parts of their usernames. For example, in the username *ObscureCoder* is reasonable to assume that the user has combined *Obscure* and *Coder*. In a more challenging example, *T0x1c V3n0m* can be split into {*T0x1c, V3n0m*} which could lead to {*Toxic, Venom*}. Note that in both these examples, we leverage the appearance of commonly-used English words, but if users use obscure geographical, or regional names, things can become more complex.

d. Token-based chunkification: We also propose an approach to detect chunks even in the absence of cues. Intuitively, we try to find potential words that could be "hiding" in the username. As an example, let us consider the following username *thegreathacker* which seems to correspond to L = {*the, great, hacker*}.[3]

There are many different ways to identify words in a string (Hall and Dowling 1980). We follow the approach below. We start from the end of the string and consider letters until we find a word that exists in our *TokenDict*, a dictionary of English word/name phrases. In our example that word would be *hacker*. We then create two parallel approaches: (a) we repeat the same process on the string, having removed *hacker*, and (b) we continue to see if the word *hacker* is part of a longer word. At the end of this process, we have several lists of *chunk*s.

4.4.4 Comparison module

Given two Bags with lists of chunks from usernames u_1 and u_2, we calculate the highest similarity score among all possible comparisons between the lists of *chunk*s. We are now ready to provide a more formal description of our approach while an intuitive visualization is provided in Fig. 6. Given usernames u_1 and u_2, earlier modules produce two sets of lists of *chunk*s, Bag(u_1) and Bag(u_2), respectively with N and M, the number of lists in each Bag.

$\text{Bag}(u_1) = [\text{L}_1^1, \text{L}_1^2, ..., \text{L}_1^N]$
$\text{Bag}(u_2) = [\text{L}_2^1, \text{L}_2^2, ..., \text{L}_2^M]$

We use three similarity functions: (a) *ChunkSim*(c_1, c_2) for *chunk*s c_1 and c_2, (b) *ListSim*(L_1, L_2) for two lists L_1 and L_2, and (c) *SimScore* (Bag_1, Bag_2) between two bags Bag_1 and Bag_2. The additional complexity is that at the list and Bag level, we need to iteratively calculate many possible potential matchings. We present a visualization of our approach in Fig. 6 using the real usernames: *z3r0c00l* and *COOL_zERO*.

a. Similarity of chunks. This is the easiest step in the similarity estimation as the similarity between strings is well established. There are many string matching algorithms, and our approach could use any of them. We select the Levenshtein method (Levenshtein et al. 1966) which is widely used in text processing (Gharibshah et al. 2020). We use the term *ChunkSim*(c_1,c_2) to refer to this function.

b. Similarity of lists of chunks. Comparing lists of chunks is slightly more complex, as we need to identify the most likely match between the *chunk*s of the two lists L_1,

[3] Though unlikely, "Thegreat" could be an unusual non-english name. A name analysis website states: "The name Thegreat is ranked on the 46,265th position of the most used names. It means that this name is rarely used. We estimate that there are at least 3500 persons in the world having this name which is around 0.001% of the population." Source: themeaningofthename.com.

L_2 (note that for clarity we drop the superscipt in the notation). We use the term *ListSim*(L_1,L_2) to refer to this function. There are many different algorithms that we can use to compare similarity of unordered lists that vary in efficiency and computational complexity. Our approach could use any such function. In our current implementation, we use the Monge-Elkan method (Monge et al. 1996) which is found to perform well consistently across many scenarios and data types (Bilenko et al. 2003) with a polynomial computational complexity of $O(|L_1||L_2|)$. Intuitively, the method iterates through the elements of the first list and identifies the highest similarity with any element in the second list. The final value is the average chunk similarity. Formally, the method calculates the similarity for the two lists as follows:

$$\text{ListSim}(L_1, L_2) = \frac{\sum_{i=1}^{|L_1|} \max_{\forall j} \ \text{ChunkSim}(L_1[i], L_2[j])}{|L_1|}$$

where $1 \leq j \leq |L_2|$ and $L_1[i]$ and $L_2[j]$ are the i-th and j-th *chunk*s of each list.

This method hides a subtle point. The function is sensitive to the order of the arguments: $\text{ListSim}(\text{L}_1, \text{L}_2) \neq \text{ListSim}(\text{L}_2, \text{L}_1)$. Therefore, one could consider three approaches. We can consider: (a) ListSim (L_1, L_2), (b) $\text{ListSim}(\text{L}_2, \text{L}_1)$, or (c) considering both "directions", $\text{ListSim}(\text{L}_1, \text{L}_2) + \text{ListSim}(\text{L}_2, \text{L}_1)$. In the results that we show here, we use one direction with the longest list as the first argument, namely, assuming $|L_1| \geq |L_2|$, $\text{ListSim}(\text{L}_1, \text{L}_2)$.

c. Similarity of Bags of lists. This is the last step in assessing the similarity between two usernames, which are represented by their Bags. We use the term Similarity Score *SimScore* (u_1,u_2) where u_1 and u_2 are the two usernames. We do the comparison exhaustively: each list in the Bag of one username is compared with each list of the other username using the list similarity function *ListSim*() from above. The Similarity Score is the maximum list similarity over all list pairs $\text{L}_1^n, \text{L}_2^m$ as follows:

$\text{SimScore}(u_1, u_2) = max_{n,m} \ \text{ListSim}(\text{L}_1^n, \text{L}_2^m)$
where $1 \leq n \leq |Bag(u_1)|, 1 \leq m \leq |Bag(u_2)|$.

4.5 Computational complexity

In our experience, the complexity was not prohibitive with most usernames having a single-digit number of lists, and each list having single-digit number of chunks. The computational complexity of the Comparison module is bounded by $O(|B_{max}|^2|L_{max}|^2)$, where $|B_{max}|$, $|L_{max}|$ are the largest Bag and list L length, respectively. The complexity of the chunkification can vary depending on the optimizations used in the exploration. Recall that we saw the distribution of chunks in Fig. 3, where 95% of usernames have less than 8 chunks.

In practice, in matching 2 M pairs of usernames, we find that it takes approximately 5 min in a Ubuntu 20.04.5 Linux machine with 64GB RAM and 8 CPU cores.

5 Experiments and evaluation

We evaluate our approach by comparing it against state of the art methods, and we demonstrate its usefulness by identifying forum users who could be malicious hackers on GitHub.

5.1 Evaluating GeekMAN

Here, we present our evaluation study and discuss our ground truth, comparison metrics, and the baseline algorithms. At a high level, our experiment aims to match a set of users from a *source* forum with a set of users from a *target* forum.

5.1.1 Baseline algorithms

The purpose of conducting an experiment is to determine how GeekMAN compares to two baseline state-of-the-art username matching methods: Wang-16 (Wang et al. 2016) and UISN-UD (Li et al. 2019).

a. Wang-16: This method extracts content features (e.g. 2-grams), and pattern features (e.g. letter-digit, date) from the usernames. Then, a vector based modeling is used to compute the cosine similarity of the vectors of features from usernames.

b. UISN-UD: This method exploits the information redundancies that can be available in a pair of usernames used by the same user. It computes features from different string comparison metrics, such as common substring, common subsequence, and edit distance, which are used in their classifier.

The major difference of GeekMAN with them is the use of deslangification and chunkification of the technogeek usernames, assuming such properties exist in the comparing usernames.

5.1.2 Experimental process and setup

We conduct two experiments focusing on: (a) technogeek and (b) regular usernames.

Experiment-T: technogeek usernames. In this experiment, we find matches between usernames from a *source* forum within a *target* forum. Specifically, we conduct two

such experiments: (a) Garage4Hackers (GH) as *source* and Offensive Community (OC) as *target*, (b) Garage4Hackers (GH) as *source* and RaidForums (RF) as *target*. Given our focus on technogeek names, we selected Garage4Hackers as *source* since it exhibits a higher level of slangified and chunkified usernames, based on our analysis of these forums (see Section 3). On the other hand, the *target* forums were picked randomly. To focus on technogeek usernames, we selected usernames from our source forum with either 3 or more *chunk*s or slangification. This way we obtain the **D_All** dataset, which is roughly 10% of the Garage4Hackers forum (on purpose small to enable validation as we discuss below). We also divide D_All into **D_Multi** (3 or more chunks) and **D_Slang** (slang conventions) datasets, in an effort to investigate the interplay between multi-chunk and slangified usernames such as {*thegreathacker, T0x1cV3n0m*}. We use GeekMAN and baselines to match each user in D_All with the most likely matching user in the *target* forums.

Experiment-R: Regular usernames. We wanted to evaluate our algorithms using regular usernames. We use a subset of the social media datasets of Table 1, which have confirmed user correspondences across the social platforms, as we explained earlier. We create a dataset **D_Social** of 98K username pairs among Google Plus (GP) and Twitter(TW) users: (a) 49K of these pairs are usernames that belong to the same user, and (b) we add 49K pairs that (most likely) do not by selecting randomly, but with a bias, among all the social usernames. To make the classification harder, we require that these false pairs have at least Levenshtein similarity matching scores of 0.3.

5.1.3 Validation and ground truth

There exists ground truth for the Experiment-R, but there is no available groundtruth for Experiment-T, so we need to establish our own. We resort to sampling and manual verification. The algorithms find the best matching user in *target* forum for the users in D_All. We recruit four domain-expert computer scientists to manually label each of these possible matches as a match or mismatch. To increase the reliability of our ground truth, we ask the annotators to match a username pair only if they were *certain* they belong to the same user based on usernames only. We only consider a username pair as a verified match if at least three annotators agree it is. Here we focus primarily on verified matches which we will use as true positives.

We assess the level of agreement of the annotators using Fleiss Kappa coefficient for labeling the ground truth matchings in Tables 3 and 4. The Kappa score we get is above 0.5, which is considered as a moderate agreement (0.41−0.60) (Emam 1999).

Table 3 Performance comparison analysis for the algorithms in D_All using $SimT = 0.7$ for GeekMAN with annotator agreement Kappa Score K=0.53

Dataset	D_All			
Method	Precision	Rel-Recall	Rel-F1-score	K
Wang-16	76.0	29.9	42.9	0.53
UISN-UD	47.4	71.6	57.1	
GeekMAN	**86.0**	62.9	**72.6**	

Table 4 Performance comparison analysis for the algorithms in D_Slang using $SimT = 0.7$ for GeekMAN with annotator agreement Kappa Score K = 0.54

Dataset	D_Slang			
Method	Precision	Rel-Recall	Rel-F1-score	K
Wang-16	77.7	25.3	38.1	0.54
UISN-UD	45.9	68.6	54.9	
GeekMAN	**81.6**	**69.9**	**75.3**	

5.1.4 The evaluation metrics

To compare our algorithms, we consider Precision, Relative Recall, and Relative F1-score for each algorithm in the experiment. The "Relative" term in the metrics , abbreviated Rel-, represents our effort to approximate the true Recall in the absence of established ground-truth. Our goal is to *detect* an algorithm that will opt for high Precision at the cost of Recall in the context of the specific comparison. We approximate the number of real matches, which we do not know in our dataset, by providing a lower bound as follows. We take the union of all true positives (validated by our annotators) of all the algorithms in the test. Formally, we define I_{algo} to be the number of matches identified by algorithm *algo*. We define TP_{algo} to be the true positives for that algorithm as verified by the annotators. We then calculate the **union of the true positives** TP_{union} as the union of all the true positives of all the algorithms: $TP_{union} = \bigcup_{a \in Algos} TP_a$. TP_{union} can be seen as a lower bound on the true matches that a perfect algorithm would have identified.

$$\text{Precision} = \frac{TP_{algo}}{I_{algo}} \qquad \text{Rel-Recall} = \frac{TP_{algo}}{TP_{union}}$$

$$\text{Rel-F1-score} = \frac{2 * \text{Precision} * \text{Rel-Recall}}{\text{Precision} + \text{Rel-Recall}}$$

As the names indicate, the Relative Recall and Relative F1-score have only relative meaning within the scope of the comparison with the specific set of algorithms.

5.1.5 Choosing the similarity score threshold

We want to identify an appropriate Similarity Score Threshold (*SimT*) value, which is a critical parameter for our approach. First, we apply GeekMAN and baseline algorithms on D_All (D_Multi + D_Slang) to find matching pairs between *source* forum and *target* forum. Second, the matchings are labeled by the annotators. Depending on their annotations, we calculate the defined evaluation metrics for our algorithm at different *SimT* values ranging between 0.1−1.0 at 0.01 intervals. Finally, we plot the Precision, Rel-Recall, Rel-F1-score curves at those values in Fig. 7. We find that the curves meet at *SimT* value of 0.64. We notice that after $SimT = 0.68$, Rel-F1-score decreases, while Precision is increasing even after 0.70. We opt to prioritize Precision, and we choose a *SimT* of 0.7 for our study.

5.1.6 Evaluation results

We show the performance evaluation for GeekMAN and the two baseline algorithms.

GeekMAN: 15% better Relative F1-score. In Table 3, we show the results for the D_All dataset. GeekMAN achieves a Precision of 86.0% and Relative F1-score of 72.6%. By contrast, the baseline algorithms achieve 76.0% and 47.4% Precision with 42.9% and 57.1% Relative F1-score, respectively. It is also worth noting that the baseline algorithms only do well either in Precision or in Relative Recall. For example, Wang-16 offers high Precision (76.0%) but at the cost of the Rel-Recall (29.9%). The opposite is true for UISN-UD.

What about bonafide technogeek names? We want to further understand how the algorithms perform when the usernames use slang conventions. As expected, our approach does even better here with a difference in the Relative F1-score close to 20%. Focusing on our D_Slang dataset, we show the results in Table 4. GeekMAN surpasses the baselines in all metrics with a Precision of 81.6% and Relative F1-score of 75.3%, while baseline algorithms achieve a maximum 77.7% Precision and 54.9% Relative F1-score.

What do the other approaches miss? We did a deepdive to understand the origin of the lower performance of the other methods. Table 5 shows some of the matching pairs identified by GeekMAN that the baselines could not identify. We conjecture that this happens due to the usernames exhibiting slangification and chunkification, like *h4ckj4ck* and *JackTheHack*. These usernames exhibit slangification and chunkification, that the baselines were not designed to handle. For some of these algorithms as they rely on standard string techniques, such as 2-grams and substring matching, that will clearly not work well here.

Table 5 Real username matches that baseline algorithms missed

Platform 1	Platform 2
c0d3m@st3r	Codomaster
41ph4w01f	alphawolfffff
h4ckj4ck	JackTheHack
OverlordShadow	Shadowlord

Table 6 Performance metrics between GeekMAN and the baseline algorithms across social network ground-truth

Dataset	D_Social		
Method	Precision	Recall	F1-Score
Wang-16	63.2	74.5	68.4
UISN-UD	66.1	75.1	70.3
GeekMAN (*SimT* =0.7)	**98.6**	52.4	**68.4**

5.1.7 GeekMAN performs well with "regular" usernames

We also show that our approach works well with regular social media usernames with Experiment-R. We apply each algorithm on D_Social to evaluate the performance. Table 6 shows that GeekMAN outperforms the baseline algorithms in terms of Precision being 98.6% with Similarity Score Threshold, SimT = 0.7, whereas the baseline algorithms have a maximum 66.1% Precision. Additionally, the F1-score of the methods is within 2% of each other.

5.2 Case study: matching GitHub malware authors

We showcase the capabilities of our approach with the following application. We use GeekMAN to identify forum users that match malware authors on GitHub (Rokon et al. 2020).

5.2.1 GeekMAN matches forum users with malware authors on GitHub

We find 6327 GitHub malware authors who have been matched with at least one user from one of our security forums. We apply our method with GitHub as the *source* platform and each of the security forums as the *target* forum. In the case that a study values Precision over Recall, we can adjust the Similarity Score Threshold. By using $SimT = 0.9$, we identify 1958 matches, which is an even stronger indication that a substantial set of these usernames may belong to the same user.

We also find 260 GitHub malware authors in the security forums using exact username matching. For comparison, Table 7 shows the number of matched forum users using GeekMAN and exact username matching. We find that GeekMAN provides 24 and 7.5 times more matching than

purely exact username matching using $SimT = 0.7$ and 0.9, respectively.

5.2.2 Deep dive: influential users and motives

Who are these malware authors and what is their activity on the security forums? Among the matched malware authors, we focus on the most influential authors who have at least 250 followers on GitHub. We then analyze the threads and posts of the counterpart matching forum users. They seem to fall into the following broad groups according to major activity: (a) promoting their GitHub repositories (e.g.*TheSph1nx, TheSphinx*), (b) requesting hacking related information (e.g. *Cr4sh, crash*), (c) sharing hacking tutorials (e.g. *Bl4ckDr460n, blackdrag0*), and (d) boasting and advertising personal hacking skills (e.g. *Anon-Exploiter, An0n3xpl0it3r*). We also detect 4 matched GitHub authors who have since had their GitHub accounts deactivated, such as author *bl4ckic3* who was promoting a keylogger repository using the username *Blackice* in forum HF.

Many malware authors (3-13%) are highly active in the forums. We wanted to see how visibly these malware authors are highly active on our security forums. We use the term **heavy-hitter** to refer to users who are in the top 10% in terms of number of posts in the forum. In Table 7, we see that 3-13% of matched malware authors are also heavy-hitters in the forum. By analyzing the URLs these users posted, we identify 18 users who share their own GitHub profile and repositories.

6 Discussion: tool and scope

We discuss the GeekMAN tool, potential scope and usecase, the limitations and impact of our approach in this section.

The GeekMAN online platform. We provide an online platform that implements our approach to enable further research in the area. A sample of our platform is shown in Fig. 1, which is publicly accessible at https://geekman.streamlit.app. Currently, we provide a set of minimal functionalities. However, in our platform, we intend to support the following capabilities incrementally.

a. Finding a match within our database(s). A user can provide a username which we will match with the top-k most similar usernames in our database. A user can specify which platforms to include in the match, e.g. specific online forums, or all of them.

b. Leveraging our matching algorithms. A user can upload two CSV files with usernames, and we can provide the best matches for each username in the first file within the second file. The user can specify which algorithms to use for the matching, and even compare the similarity scores for the algorithms.

c. Providing the largest username database, ground truth and slang-dictionary. We intend to develop the largest database of usernames by adding more platforms that focus on technogeek space. We will also provide our manually verified results as groundtruth, which we expect to grow over time with the help of experts and maybe even using carefully-run MTurk studies. Finally, we will share our technogeek dictionary which already has 5.8K entries.

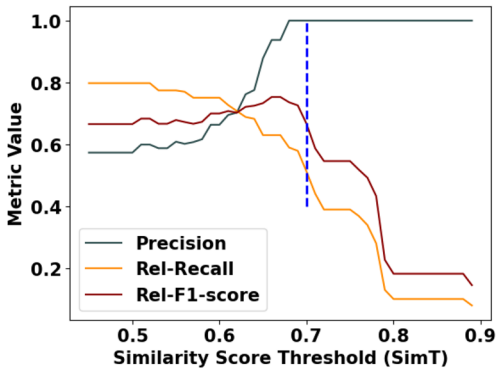
Scope, practical applications and impact. We see our approach as a key building block for tracing malicious users across online platforms. Its focus on technogeek usernames makes it particularly well suited for the disambiguation of such users which, apart from malicious hackers, could include hate groups, and sociopaths, which could be of interest to law enforcement entities. GeekMAN amplifies the ability of security analysts to quickly hone in on *persons of interest* and comprehensively track them across platforms. We see it as the first step in the following pipeline: (a) collecting usernames from forums of interest, (b) identifying likely similar usernames, (c) investigating further with more computationally-expensive methods, and (d) involving humans in the effort. The goal is to enable efforts to stop individuals that cause real harm through cyber-criminal activities, such as ransomware, Denial of Service attacks, and identify theft. We would like to stress that a username match should be treated carefully as we discuss below.

How representative is our data? This is the question that plagues any measurement-driven effort. We argue that our data is sufficiently representative for the purposes of the study. The security forums that we use are among the appropriate targets that a security analyst would examine: they are highly technical and in the grey area between white and black hat hacking. Our successful linking of malware authors from GitHub to users in our forums is an additional indication of using relevant datasets. Our goal is to show the challenges and offer a solution in disambiguating technogeek usernames.

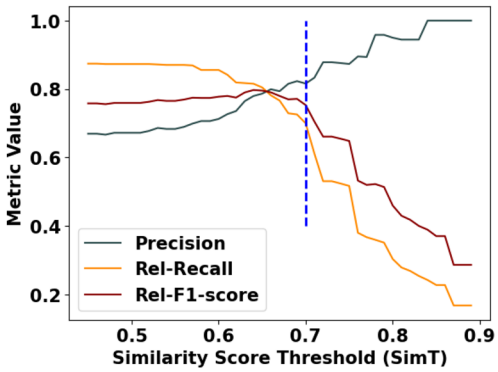
Table 7 GeekMAN finds more forum-GitHub matches ($SimT = 0.7$) compared to exact matches alone

Forum	GitHub author matched		Heavy-hitters (%)
	Exact match	GeekMAN	
GH	3	226	13.2
OC	16	1306	3.7
RF	20	2553	7.3
MP	64	4767	10.9
HF	182	5111	9.9

A substantial percentage of them are heavy-hitters in that forum. Note that a malware author can match users in more than one forum



(a) D_Multi: Precision, Rel-Recall, Rel-F1-score.



(b) D_Slang: Precision, Rel-Recall, Rel-F1-score.

Fig. 7 The effect of *SimT* on the performance of GeekMAN. The blue vertical line at 0.7 marks the point that finds a reasonable trade-off with a slight emphasis on Precision

Limitations and extensions. We discuss limitations and potential extensions of the work.

Username vs. identity. It should be clear that identifying similar or even identical usernames does not guarantee that the owners behind them are the same person. However, we argue that this is a good place to start in absence of other leads. Our automatic sifting through hundreds of thousands or millions of usernames is essential to make such an investigation feasible.

Expanding the matching capability. We intend to further develop the matching capability by considering additional sources of information when these are available. For example, we will consider (a) the content of the posts, (b) posted or linked URLs, and (c) social connectivity. For the latter, we can extract "social links" by finding groups of users that post in the same threads, or that refer to each others' resources and accounts on YouTube and GitHub.

Extensions. We intend to scale up our study by: (a) collecting data from more security forums and (b) interconnecting their users with other platforms, including YouTube, which provides hacking tutorials, communication platforms like Telegram and Discord, and social media, which are known to harbor hackers (from personal communication with Amazon's security team).

Ethical considerations. Our work adheres to the community-defined ethical practices. First, we only use publicly available data. Anyone can collect the data from the forums and we use other data that has been made public by other researchers. Second, the usernames are not directly revealing any particular individual, but only represent an online persona. We only use specific usernames to illustrate naming conventions but we will be happy to obfuscate or use artificial examples in the final version of the journal. Finally, we want to stress: a pair of matching usernames simply points to the probability that there is a connection and further proof should be solicited before any action is taken.

7 Related work

Most previous works differ from our approach in that: (a) they are supervised approaches that need training data, (b) they are not focusing on complex-technogeek usernames, and (c) they rely on information beyond the username, such as user profile attributes, content, and social connectivity. By contrast, our approach is designed to: (a) handle technogeek names, and (b) rely only on usernames.

a. Username-based matching. We already discussed the two methods that we use in our performance analysis; (Wang et al. 2016) and (Li et al. 2019).

Earlier, Perito et al. (2011) employed a Markov-Chain based language model to quantify username uniqueness. The authors suggest that the selection of usernames follows a probabilistic distribution, with more predictable and less diverse usernames being more likely to be duplicated. To quantify this probability, they estimate the probability of a username's occurrence based on a large username dataset leveraging Markov Chains model. To measure the similarity between usernames, they employ distance metrics such as Levenshtein distance and Jaro distance, alongside term frequency-inverse document frequency (TF-IDF).

Other efforts by Zafarani and Liu (2013) proposed MOBIUS to identify users across sites by analyzing the naming patterns of usernames. This technique exploits the redundant information in username patterns revealed through user behavior. MOBIUS considers various user behavioral patterns, such as keyboard layouts, language usage, and username modification habits, to construct features. To link

usernames, it employs several similarity measures, including Longest Common Substring (LCS), edit distance, Dynamic Time Warping distance, Jensen-Shannon divergence, and n-gram algorithm.

However, both previous approaches did not address the unique characteristics of usernames in the technogeek forum context as GeekMAN does. Additionally, technogeek users frequently engage as silent observers in the forum, leading to the unavailability of their behavioral patterns. This poses a challenge for earlier username disambiguation methods that rely on such data.

b. Using user profiles. An earlier study by Vosecky et al. (2009) introduced a vector based supervised algorithm which utilizes user profile features with differing weights. In some other efforts, Goga et al. (2013) and (Zhang et al. 2014 proposed probabilistic classifiers which link user identities based on profile attributes like description, location, profile image etc including username. Also, a projection based modeling proposed by Mu et al. (2016) incorporated the profile features to link users on different platforms.

c. Combining multiple types of information. A number of efforts like (Malhotra et al. 2012; Zhou et al. 2015; Zafarani et al. 2015; Zhang et al. 2015; Liu et al. 2016 consider the social relationship for user matching. On the other hand, Liu et al. (2013); Jain et al. (2013); Goga et al. (2015); Arabnezhad et al. (2020); Cabrero-Holgueras and Pastrana (2021) incorporate a set of user related features including profile attributes, social relationships, and user generated contents. Some recent approaches (Zhang et al. 2018, 2019 leverage a user's ego network to connect users across platforms.

8 Conclusion

We propose GeekMAN, a systematic approach to identify similar technogeek usernames across online platforms. The key novelty consists of the development and integration of three capabilities: (a) decomposing usernames into meaningful chunks, (b) de-obfuscating technical and slang conventions, and (c) considering all the different outcomes of the two previous functions exhaustively when calculating the similarity. These three capabilities attempt to emulate the way a human will attempt to "understand" a username. We conduct a study using 1.8M usernames from three different types of forums: (a) security forums, (b) malware authors from GitHub, and (c) mainstream social media platforms. Our experiment shows that GeekMAN can match technogeek usernames with offering a higher precision of 81.6% and above, outperforming the previous studies. Besides, our approach outperforms prior methods even with usernames of common social networks in terms of precision. In the future, we intend to apply our method in other varied forums and platforms to interconnect users. Overall, we see our approach as fundamental building block for tracing users across different platforms by providing: (a) our platform, (b) datasets, (c) groundtruth, and (d) a slang-translation dictionary.

Acknowledgments This work was supported by NSF SaTC grant No. 2132642.

References

(2022) https://www.cambridgecybercrime.uk
Arabnezhad E, La Morgia M, Mei A, Nemmi EN, Stefa J (2020) A light in the dark web: Linking dark web aliases to real internet identities. In: ICDCS
Bilenko M, Mooney R, Cohen W, Ravikumar P, Fienberg S (2003) Adaptive name matching in information integration. IEEE Intell Syst
Cabrero-Holgueras J, Pastrana S (2021) A methodology for large-scale identification of related accounts in underground forums. Comput Secur
Community O (2021) http://offensivecommunity.net
Emam KE (1999) Benchmarking kappa: Interrater agreement in software process assessments. Empirical Software Engineering
FBI (2020) https://www.fbi.gov/wanted/cyber/behzad-mohammadzadeh
Garage4Hackers (2021) http://garage4hackers.com
GeekMAN (2023) https://github.com/mrayhanulmasud/geekman
Gharibshah J, Papalexakis EE, Faloutsos M (2018) RIPEx: Extracting malicious ip addresses from security forums using cross-forum learning. In: PAKDD
Gharibshah J, Papalexakis EE, Faloutsos M (2020) REST: A thread embedding approach for identifying and classifying user-specified information in security forums. In: ICWSM
Goga O, Loiseau P, Sommer R, Teixeira R, Gummadi KP (2015) On the reliability of profile matching across large online social networks. In: ACM SIGKDD
Goga O, Perito D, Lei H, Teixeira R, Sommer R (2013) Large-scale correlation of accounts across social networks. University of California at Berkeley, Berkeley, California, Tech Rep TR-13-002
Hackforums (2021) https://hackforums.net
Hacking MG (2021) https://www.mpgh.net
Hall PA, Dowling GR (1980) Approximate string matching. ACM computing surveys (CSUR)
Islam R, Rokon MOF, Darki A, Faloutsos M (2021a) HackerScope: The dynamics of a massive hacker online ecosystem. Social Network Analysis and Mining
Islam R, Rokon MOF, Papalexakis EE, Faloutsos M (2021b) Recten: A recursive hierarchical low rank tensor factorization method to discover hierarchical patterns from multi-modal data. In: ICWSM
Islam R, Rokon MOF, Papalexakis EE, Faloutsos M (2022) Tenfor: Tool to mine interesting events from security forums leveraging tensor decomposition. In: Social Media Analysis for Event Detection, Springer, pp 57–87
Jain P, Kumaraguru P, Joshi A (2013) @ i seek'fb. me' identifying users across multiple online social networks. In: WWW
Levenshtein VI, et al. (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady
Li Y, Peng Y, Zhang Z, Yin H, Xu Q (2019) Matching user accounts across social networks based on username and display name. World Wide Web
Liu L, Cheung WK, Li X, Liao L (2016) Aligning users across social networks using network embedding. In: IJCAI

Liu J, Zhang F, Song X, Song YI, Lin CY, Hon HW (2013) What's in a name? an unsupervised approach to link users across communities. In: WSDM
Malhotra A, Totti L, Meira Jr W, Kumaraguru P, Almeida V (2012) Studying user footprints in different online social networks. In: ASONAM
Monge AE, Elkan C, et al. (1996) The field matching problem: algorithms and applications. In: KDD
Mu X, Zhu F, Lim EP, Xiao J, Wang J, Zhou ZH (2016) User identity linkage by latent user space modelling. In: ACM SIGKDD
Perito D, Castelluccia C, Kaafar MA, Manils P (2011) How unique and traceable are usernames? In: International Symposium on Privacy Enhancing Technologies Symposium, Springer
RaidForums (2021) https://raidforums.com
Rokon MOF, Islam R, Darki A, Papalexakis EE, Faloutsos M (2020) SourceFinder: Finding malware source-code from publicly available repositories in github. In: RAID
Samtani S, Chen H (2016) Using social network analysis to identify key hackers for keylogging tools in hacker forums. In: Intelligence and Security Informatics (ISI), IEEE
SourceFinder (2022) http://www.source-finder.org
Vosecky J, Hong D, Shen VY (2009) User identification across multiple social networks. In: First International Conference on Networked Digital Technologies, IEEE
Wang Y, Liu T, Tan Q, Shi J, Guo L (2016) Identifying users across different sites using usernames. Procedia Computer Science
Wikipedia (2023) https://en.wikipedia.org/wiki/Leet
Zafarani R, Liu H (2013) Connecting users across social media sites: a behavioral-modeling approach. In: ACM KDD
Zafarani R, Tang L, Liu H (2015) User identification across social media. ACM TKDD
Zhang J, Chen B, Wang X, Chen H, Li C, Jin F, Song G, Zhang Y (2018) Mego2vec: Embedding matched ego networks for user alignment across social networks. In: ACM CIKM
Zhang F, et al L (2019) Oag: Toward linking large-scale heterogeneous entity graphs. In: ACM SIGKDD
Zhang H, Kan MY, Liu Y, Ma S (2014) Online social network profile linkage. In: Asia Information Retrieval Symposium, Springer
Zhang Y, Tang J, Yang Z, Pei J, Yu PS (2015) COSNET: Connecting heterogeneous social networks with local and global consistency. In: ACM SIGKDD
Zhou X, Liang X, Zhang H, Ma Y (2015) Cross-platform identification of anonymous identical users in multiple social media networks. IEEE Trans Knowl Data Eng