GeekMAN: Geek-oriented username Matching Across online Networks

Md Rayhanul Masud
UC Riverside
mmasu012@ucr.edu

Ben Treves
UC Riverside
btrev003@ucr.edu

Michalis Faloutsos
UC Riverside
michalis@cs.ucr.edu

***Abstract*—How can we identify malicious hackers participating in different online platforms using their usernames only? Disambiguating users across online platforms (e.g. security forums, GitHub, YouTube) is an essential capability for tracking malicious hackers. Although a hacker could pick arbitrary names on different platforms, 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 1.2M usernames from five security forums. Our method outperforms previous methods with a Precision of 81-86%. We see our approach as a fundamental research capability, which we made publicly available on GitHub.**

I. 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 [1]. Second, these hackers are active and visible on many online platforms including specialized security forums and popular platforms like GitHub [2]. In fact, some of these platforms harbor malicious activities to the point that they are forced to shut down [3]. 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.

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 goal is tracking hackers, 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, a username of interest could be *w33dgod*, which we may want to match with

ASONAM '23, November 6–9, 2023, Kusadasi, Turkiye
© 2023 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0409-3/23/11.
https://doi.org/10.1145/3625007.3627498

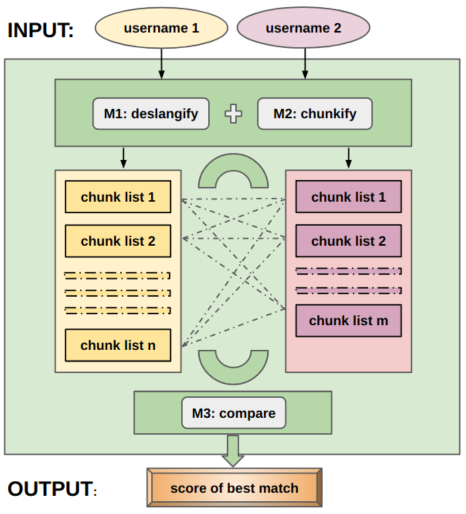


Fig. 1: GeekMAN calculates the similarity score for a pair of usernames focusing on technogeek users. Our approach tries to emulate human interpretation by combining the: (a) Chunkification, (b) Deslangification, and (c) Comparison modules.

godweed (both are real usernames). 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 matching accuracy.

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. As we explain later, we compare our approach against a set of state-of-the-art username similarity algorithms [4], [5]. We discuss previous works in Section V.

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 *B14CKH4K3R*. The key novelty of our work consists of the development and integration of three capabilities: (a) **deslangification**,

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

TABLE I: Summary of the dataset.

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 deploy our approach on 1.2M usernames from five popular hacker-rich security forum users. The key results are summarized below.

a. Technogeek usernames use slang and chunks extensively. We find that 17-37% of the technogeek platform users use multiple digits in their usernames. Quantifying the prevalence of chunks, we notice that 60-70% of the usernames could be decomposed into at least four chunks.

b. GeekMAN outperforms prior approaches. 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 is a fundamental building block for user disambiguation with an emphasis on technogeek usernames and we made our code available on GitHub [6].

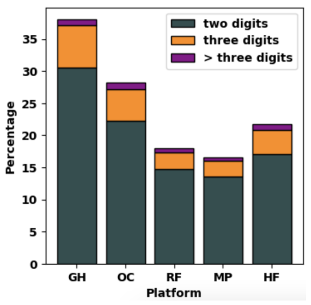
II. Background and Data

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

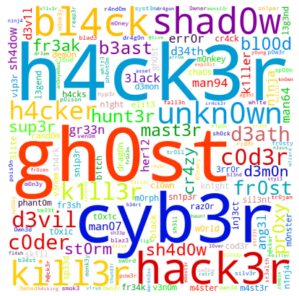
A. Malicious hackers use technogeek usernames. Hackers and other malicious users often participate in various public platforms, including specialized discussion forums [7], technical forums, and software platforms like GitHub [8]. Their main goals seem to be: (a) establishing an online brand [9] and (b) boasting of their accomplishments [2], [10]. 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 [11].

B. Datasets. Our dataset summarized in Table I contains five online security forums: Garage4Hackers(GH), Offensive Community(OC), RaidForums(RF), Multiplayer Game Hacking(MP), and Hack Forums(HF) [12], [13], [14], [15], [16]. 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[17], who kindly shared their data with us.

C. Quantifying technogeek usernames. We study the usernames from our technogeek forums which are likely to be visited by hackers. We plot the distribution of the usernames having multiple digits in between letters of their usernames in Figure 2(a). We see that around 17-37% of the usernames show the behavior. For example: *n1nj4sec* and *z3r0d4y* contain two and three digits in between letters, instead of ninjasec and



(a) Digit usage in usernames



(b) Popular slangified words

Fig. 2: (a) Distribution of percentage of population using digit in between letters multiple times. (b) The word-cloud that shows popular slangified words used as usernames or as parts of usernames.

zeroday. We illustrate the most commonly slangified English words used by technogeek forum users via a word cloud shown in Figure 2(b). In addition, we find that approximately 60-70% of the users on each of the online platforms contain 4 or more chunks in their usernames.

Validation and groundtruth. We describe our validation approach in Section IV.

III. Proposed Method

The goal of our method is to determine the similarity of two usernames deriving inspiration from a human interpretation. The key idea is to de-obfuscate the usernames (if possible), decompose them into one or more lists of chunks, and compare all possible lists of chunks to calculate the similarity score leveraging three main modules: (a) Deslangification, (b) Chunkification, and (c) Comparison, which we discuss below. We provide a conceptual overview in Figure 1 and demonstrate its operation for usernames *z3r0c00l* and *COOL_zERO* in Figure 3.

A. Maximizing the likelihood of a match. The task of reverse engineering the naming habits of users is a challenging problem. 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

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

B. Deslangification module. In this module, we try to de-obfuscate the username to identify any available slangified $chunk$. First, we create a function, $slangCharMap()$, that maps a potential slang character to a letter following the common technogeek conventions based on our observations and commonly reported usage [18]. 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 the username.

C. 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.

a. Symbol-based chunkification: Symbols are used as delimiters to chunkify usernames. These symbols are the underscore, the space, and the dot. We create L, a list of $chunk$s using the symbols present in the username. An example is *COOL_zERO* which is split into {*COOL, zERO*}.

b. Digit-based chunkification: Digits in a username can be either something telling about the user or a user may also 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; e.g. the username *ObscureCoder* is reasonable to assume that the user has combined *Obscure* and *Coder*. In a more challenging example, *T0x1cV3n0m* can be split into {*Toxic, Venom*}. Note that here, we leverage the commonly-used English words, but if users use obscure geographical/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. As an example, let us consider the following username *thegreathacker* which seems to correspond to L = {*the, great, hacker*}. There are many different ways to identify words within a string [19]. We follow the approach: 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.

D. 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. 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* (u_1, u_2) between two bags Bag_1 and Bag_2.

a. Similarity of chunks. There are many string matching algorithms, and our approach could use any of them. We select the Levenshtein method [20] which is widely used in text processing [21]. 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, 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

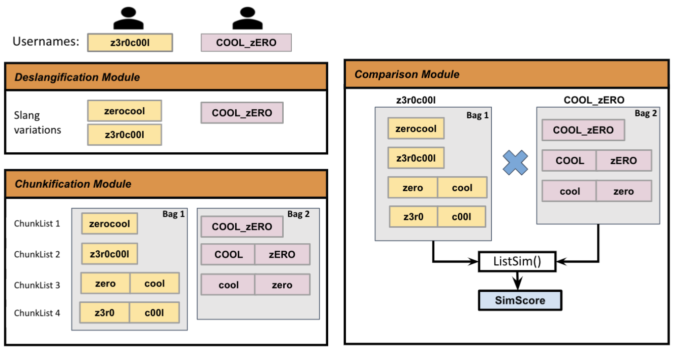


Fig. 3: An example of how the modules of our approach will handle a pair of usernames: *z3r0c00l* and *COOL_zERO*.

complexity. Our approach could use any such function. In our current implementation, we use the Monge-Elkan method [22] which is found to perform well consistently across many scenarios and data types [23] 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:

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

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, respectively.

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

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

$$SimScore(u_1, u_2) = max_{n,m}\ \ ListSim(\text{L}_1^n, \text{L}_2^m)$$

where $1 \leq n \leq |Bag(u_1)|$, $1 \leq m \leq |Bag(u_2)|$.

IV. Experiments and Evaluation

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.

A. Baseline. We evaluate our approach by comparing it against state of the art methods.

a. Wang-16 [4]: 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 [5]: 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 deslangification and chunkification of the technogeek usernames, since such properties exist in technogeek usernames.

B. Experimental setup. 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 $target$, and (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 II). 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.

C. Validation and ground truth. Since there is no available groundtruth, 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 Table II and Table III. The Kappa score we get is above 0.5, which is considered as a moderate agreement (0.41-0.60) according to standard practice [24].

D. 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 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.

E. 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, and Rel-F1-score curves at those values. 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 increases even after 0.70. We opt to prioritize Precision, and we choose a *SimT* of 0.7 for our study.

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

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 II: Performance comparison analysis for the algorithms in D_All using *SimT* = 0.7 for GeekMAN with Kappa Score K=0.53.

GeekMAN: 15% better Relative F1-score. In Table II, 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.

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**	

TABLE III: Performance comparison analysis for the algorithms in D_Slang using *SimT* = 0.7 for GeekMAN with Kappa Score K=0.54.

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 III. 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.

V. 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-16 [4] and UISN-UD [5]. Earlier, Perito et al. [25] estimated the uniqueness of a username using a Markov-Chain based language model to compute username similarity. Other efforts by Zafarani et al. [26] exploited the redundant information available in username patterns exposed by user behaviors.

b. Using user profiles. An earlier study by Vosecky et al. [27] introduced a vector based supervised algorithm which utilizes user profile features with differing weights. In some other efforts, Goga et al. [28] and Zhang et al. [29] 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. [30] incorporated the profile features to link users on different platforms.

c. Combining multiple types of information. A number of efforts like [31], [32], [33], [34], [35] consider the social relationship for user matching. On the other hand, [36], [37], [38], [39], [40] combine information including profile attributes, social relationships, and user generated contents.

VI. 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. Overall, we see our approach as a fundamental building block for linking technogeek users across different platforms.

VII. Acknowledgment

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

References

[1] S. Samtani and H. Chen, "Using social network analysis to identify key hackers for keylogging tools in hacker forums," in *ISI, IEEE*, 2016.

[2] R. Islam, M. O. F. Rokon, A. Darki, and M. Faloutsos, "HackerScope: The dynamics of a massive hacker online ecosystem," *SNAM*, 2021.

[3] J. Gharibshah, E. E. Papalexakis, and M. Faloutsos, "RIPEx: Extracting malicious ip addresses from security forums using cross-forum learning," in *PAKDD*, 2018.

[4] Y. Wang, T. Liu, Q. Tan, J. Shi, and L. Guo, "Identifying users across different sites using usernames," *Procedia Computer Science*, 2016.

[5] Y. Li, Y. Peng, Z. Zhang, H. Yin, and Q. Xu, "Matching user accounts across social networks based on username and display name," *World Wide Web*, 2019.

[6] GeekMAN, 2023. [Online]. Available: https://github.com/mrayhanulmasud/geekman

[7] B. Treves, M. R. Masud, and M. Faloutsos, "URLytics: Profiling forum users from their posted urls," in *ASONAM*. IEEE, 2022.

[8] M. O. F. Rokon, R. Islam, M. R. Masud, and M. Faloutsos, "PIMan: A comprehensive approach for establishing plausible influence among software repositories," in *ASONAM*. IEEE, 2022.

[9] E. Mariconti, J. Onaolapo, S. S. Ahmad, N. Nikiforou, M. Egele, N. Nikiforakis, and G. Stringhini, "What's in a name? understanding profile name reuse on twitter," in *World Wide Web*, 2017.

[10] R. Islam, M. O. F. Rokon, E. E. Papalexakis, and M. Faloutsos, "Recten: A recursive hierarchical low rank tensor factorization method to discover hierarchical patterns from multi-modal data," in *ICWSM*, 2021.

[11] FBI, 2020. [Online]. Available: https://www.fbi.gov/wanted/cyber/behzad-mohammadzadeh

[12] Garage4Hackers, 2021. [Online]. Available: http://garage4hackers.com

[13] O. Community, 2021. [Online]. Available: http://offensivecommunity.net

[14] RaidForums., 2021. [Online]. Available: https://raidforums.com

[15] M. G. Hacking, 2021. [Online]. Available: https://www.mpgh.net

[16] Hackforums, 2021. [Online]. Available: https://hackforums.net

[17] CambridgeCybercrimeCentre, 2022. [Online]. Available: https://www.cambridgecybercrime.uk

[18] Wikipedia, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Leet

[19] P. A. Hall and G. R. Dowling, "Approximate string matching," *ACM computing surveys (CSUR)*, 1980.

[20] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, 1966.

[21] J. Gharibshah, E. E. Papalexakis, and M. Faloutsos, "REST: A thread embedding approach for identifying and classifying user-specified information in security forums," in *ICWSM*, 2020.

[22] A. E. Monge, C. Elkan *et al.*, "The field matching problem: algorithms and applications." in *KDD*, 1996.

[23] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration," *IS*, 2003.

[24] K. E. Emam, "Benchmarking kappa: Interrater agreement in software process assessments," *Empirical Software Engineering*, 1999.

[25] D. Perito, C. Castelluccia, M. A. Kaafar, and P. Manils, "How unique and traceable are usernames?" in *PETS*. Springer, 2011.

[26] R. Zafarani and H. Liu, "Connecting users across social media sites: a behavioral-modeling approach," in *ACM KDD*, 2013.

[27] J. Vosecky, D. Hong, and V. Y. Shen, "User identification across multiple social networks," in *Networked Digital Technologies*. IEEE, 2009.

[28] O. Goga, D. Perito, H. Lei, R. Teixeira, and R. Sommer, "Large-scale correlation of accounts across social networks," *University of California at Berkeley, Berkeley, California, Tech. Rep. TR-13-002*, 2013.

[29] H. Zhang, M.-Y. Kan, Y. Liu, and S. Ma, "Online social network profile linkage," in *Asia Information Retrieval Symposium*. Springer, 2014.

[30] X. Mu, F. Zhu, E.-P. Lim, J. Xiao, J. Wang, and Z.-H. Zhou, "User identity linkage by latent user space modelling," in *ACM SIGKDD*, 2016.

[31] A. Malhotra, L. Totti, W. Meira Jr, P. Kumaraguru, and V. Almeida, "Studying user footprints in different online social networks," in *ASONAM*, 2012.

[32] X. Zhou, X. Liang, H. Zhang, and Y. Ma, "Cross-platform identification of anonymous identical users in multiple social media networks," *IEEE transactions on knowledge and data engineering*, 2015.

[33] R. Zafarani, L. Tang, and H. Liu, "User identification across social media," *ACM TKDD*, 2015.

[34] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "COSNET: Connecting heterogeneous social networks with local and global consistency," in *ACM SIGKDD*, 2015.

[35] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding." in *IJCAI*, 2016.

[36] J. Liu, F. Zhang, X. Song, Y.-I. Song, C.-Y. Lin, and H.-W. Hon, "What's in a name? an unsupervised approach to link users across communities," in *WSDM*, 2013.

[37] P. Jain, P. Kumaraguru, and A. Joshi, "@ i seek'fb. me' identifying users across multiple online social networks," in *WWW*, 2013.

[38] O. Goga, P. Loiseau, R. Sommer, R. Teixeira, and K. P. Gummadi, "On the reliability of profile matching across large online social networks," in *ACM SIGKDD*, 2015.

[39] E. Arabnezhad, M. La Morgia, A. Mei, E. N. Nemmi, and J. Stefa, "A light in the dark web: Linking dark web aliases to real internet identities," in *ICDCS*, 2020.

[40] J. Cabrero-Holgueras and S. Pastrana, "A methodology for large-scale identification of related accounts in underground forums," *Computers & Security*, 2021.