Mitigating Colluding Attacks in Online Social Networks and Crowdsourcing Platforms

Georges Arsene K. Kamhoua
Florida International University, gkamh001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

MITIGATING COLLUDING ATTACKS IN ONLINE SOCIAL NETWORKS AND CROWDSOURCING PLATFORMS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Georges Arsène Kamhoua Kamto

2019

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Georges Arsène Kamhoua Kamto, and entitled Mitigating Colluding Attacks in Online Social Networks and Crowdsourcing Platforms, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sundaraja Sitharama Iyengar

Deng Pan

Jean H. Andrian

Leonardo Bobadilla

Laurent Y. Njilla

Niki Pissinou, Major Professor

Date of Defense: June 28, 2019

The dissertation of Georges Ars`ene Kamhoua Kamto is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2019

DEDICATION

To my family.

ACKNOWLEDGMENTS

First, I would like to express my gratitude to my advisor, Dr. Niki Pissinou, for supporting me for these past four years. She has given me direction, advice, and support from the beginning until the completion of this work throughout my Ph.D. at the Florida International University. I am also thankful to her for her encouraging advice, knowledge, insightful guidance and discussions with longstanding patience. Her training taught me to always look up the “Why” question of science, help me become an independent researcher, and overcome significant challenges.

I am thankful for all my thesis committee members: Dr. Sundaraja Sitharama Iyengar, Dr. Deng Pan, Dr. Jean Andrian, Dr. Leonardo Bobadilla, and Dr. Laurent Njilla for their time taken to serve on my dissertation committee. Their comments and valuable feedback were instructive and helped me with the quality of this work. I would like to specially thanks Dr. Laurent Njilla for his collaboration, unlimited patient, and suggestions during my Ph.D. study.

I would like to thank Dr. Charles A. Kamhoua of Army Research Laboratory for the interactive research discussions that we frequently had.

Additionally, I also thank the excellent faculty members of the School of Computing and Information Sciences for their lectures and projects. Thank you to Colonel Jerry Miller, Mrs. Olga Carbonell and Mrs. Ariana Taglioretti.

I want to acknowledge my lab mates at Telecommunication and Information Technology Institute (IT2). Especially thanks go to Dr. Mingming Guo, Concepción Sánchez Alemán, Abdur Rahman Bin Shahid, Samia Tasnim, Hussein Zangoti, and Ricardo Boetto for their constant motivation and supports. Furthermore, parts of the results in a few chapters were joint work with the undergraduate student Jonathan Beltran, Navneedh Maudgalya, and Benjamin Liang I have mentored in NSF/REU program.

I am thankful to have close friends like Jean Hervé Kolle, Yves Kamole, Gerardin Kamdem, Emmanuel Lokolo, Aziz Betche, and Christian Djiofack. I am happy for the support they have provided.

Furthermore, I express my deepest gratitude and love for my family members for the stimulation and support they have provided me during this journey. Especially to my mother, Mamko Solange, who has always encouraged me to pursue my dream. Also, I would like to appreciate all the devotion and sacrifice made by Kimberly Martin during my dissertation.

Finally, I would like to acknowledge that my graduate studies have been partially funded by the US National Science Foundation, Department of Defense, the graduate school at Florida International University through the Dissertation Year Fellowship, and the School of Computing and Information Sciences.

ABSTRACT OF THE DISSERTATION

MITIGATING COLLUDING ATTACKS IN ONLINE SOCIAL NETWORKS AND CROWDSOURCING PLATFORMS

by

Georges Arsène Kamhoua Kamto

Florida International University, 2019

Miami, Florida

Professor Niki Pissinou, Major Professor

Online Social Networks (OSNs) have created new ways for people to communicate, and for companies to engage their customers – with these new avenues for communication come new vulnerabilities that can be exploited by attackers. This dissertation aims to investigate two attack models: Identity Clone Attacks (ICA) and Reconnaissance Attacks (RA). During an ICA, attackers impersonate users in a network and attempt to infiltrate social circles and extract confidential information. In an RA, attackers gather information on a target's resources, employees, and relationships with other entities over public venues such as OSNs and company websites. This was made easier for the RA to be efficient because well-known social networks, such as Facebook, have a policy to force people to use their real identities for their accounts. The goal of our research is to provide mechanisms to defend against colluding attackers in the presence of ICA and RA collusion attacks. In this work, we consider a scenario not addressed by previous works, wherein multiple attackers collude against the network, and propose defense mechanisms for such an attack. We take into account the asymmetric nature of social networks and include the case where colluders could add or modify some attributes of their clones. We also consider the case where attackers send few friend requests to uncover their targets.

To detect fake reviews and uncovering colluders in crowdsourcing, we propose a semantic similarity measurement between reviews and a community detection algorithm to overcome the non-adversarial attack. ICA in a colluding attack may become stronger and more sophisticated than in a single attack. We introduce a token-based comparison and a friend list structure-matching approach, resulting in stronger identifiers even in the presence of attackers who could add or modify some attributes on the clone. We also propose a stronger RA collusion mechanism in which colluders build their own legitimacy by considering asymmetric relationships among users and, while having partial information of the networks, avoid recreating social circles around their targets. Finally, we propose a defense mechanism against colluding RA which uses the weakest person (e.g., the potential victim willing to accept friend requests) to reach their target.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

1.1 Background

In this dissertation, we address the security problem that users or organizations can encounter while utilizing Online Social Networks (OSNs) and crowdsourcing platforms. Online social networks are public social platforms that offer users services such as connecting with remote individuals, sharing of activities and experiences, and, most notably, sharing personal information. With the help of OSNs, social circles expand from family, friends, and colleagues to communities with similar interests and preferences, regardless of their geographical location. OSNs became a fundamental component in people's daily life because some users view them not only as a platform to establish contacts but also as a means of soliciting contributions for personal and charitable causes. Most OSNs encourage users to provide large amounts of personal data, such as relationship status, phone numbers, address, their current location, share information regarding their everyday lives, opinions, political affiliations, etc. The information provided helps ingrain users in the network and drives traffic, but creates treasure troves of information for hackers [JCW$^+$13], making OSNs a valuable target for malicious users who harvest personal data for conducting attacks in the future. Hence, OSNs became a leading weapon for launching cyberattacks on organizations and individuals [Cis18].

By hacking into accounts of well-connected users, hackers can post false information with the intention to cause economic damages or create tension among people or nations. Notable examples of this type of cyberattack include the oil price spikes in 2012 because of a malicious tweet feed on the Syrian President's death [Cis18]. In 2015, supporters of the Islamic State hacked U.S. Central Command

Twitter account to post pro-ISIS messages and information related to U.S. military officials [Ros15]. Later that year, U.S. State Department officials had their social media accounts taken over by Iranian hackers to harvest information about issues the U.S. has against Iran after both countries signed a nuclear accord [SP15]. The previously enumerated examples showed how information shared in online platforms may impact everyone's real life. Because of the large amounts of information that is shared in OSNs, hackers can gain intimate knowledge of the daily lives of OSN users. Another aspect to the non malicious use of those valuable personal data was seen when the Director of National Intelligence signed a new directive policy in May 2016 that allows investigators to gather publicly available information on social media to profile people who are undergoing background checks for security clearance. [Aft18].

Another aspect of this dissertation focuses on crowd problem-solving. Online reviews via crowdsourcing systems such as Amazon Mechanical Turk (AMT) have become one of the most common ways for organizations to gather ideas for new products and services from large crowds of consumers by offering monetary rewards depending on the tasks. Many businesses rely on reviews to make significant decisions on their services and operations. On amazon mechanical turk, each complete Human Intelligence Task (HIT) by any user corresponds to financial gain accordingly. HIT tasks can include classifying images or videos, identifying objects in a photo or video matching data records, classifying website content, or collecting data about the real world, giving reviews on different products, etc. Usually, organizations that rely on reviews to make business decision assume they are written by legitimate users who are sharing their truthful opinions based on their experiences with the products. Therefore we need to ask: Are these reviews always genuine? Unfortunately not. This monetary reward has attracted malicious users who wish to complete the task with minimal effort through collaboration. For instance, a task based on reviews is degraded when malicious users copy each other with minimal edits of the review, giving a misrepresentation of the true quality of the product [KCJ14].

1.2 Motivation

OSNs have caused an explosion of online human interaction and connectivity. Each minute 49,380 posts are created on Instagram and 473,400 tweets are shared on Twitter [Ahm18]. Facebook is one of the largest and most used OSNs, and it involves 1.52 billion daily active users and 2.32 billion monthly active users on January 30, 2019 [Fac19]. Among the exchanged information, we have sensitive ones (e.g., health conditions, relationship status, family details, address, political affiliations, workplace information) that users only shared with their inner trust circle. Therefore, information leaks are disastrous for companies, organizations, and individuals whose data can be skillfully exploited. Moreover, the attackers could be malicious users (e.g., cyber espionage, cyber warfare, advertisers for financial gains, and hacktivism) interested in sensitive information [ZG16]. However, security research proposed to cope with malicious users in online social networks and crowdsourcing platforms mostly focused on single attackers. A colluding attack can be defined as an attack that includes many malicious users intending to obtain a better benefit (e.g., gather more information on a victim user) by working as a group rather than independently launched attacks. The collusion attack is one of the strongest types of behavior collaborative malicious users can adopt in the aim to evade existing defense mechanisms which cope with only single attackers in the networks [LL16, LL18]. These limitations motivate our research in new methodologies to mitigate malicious users involve in the colluding attack. In addition, it has been observed that each malicious user could present different behavior (e.g., activity pattern, connectivity pattern) than other members involved in a collusion attack. Furthermore, for attackers involved in a collusion attack, the objective is not for all attackers to necessarily succeed; regardless of whether if individual attack is successful, attackers may be able to collectively scrape sufficient information. Therefore, different collaborative malicious behaviors related to colluding attacks are proposed and evaluated in this research, for the purpose of developing new defense mechanisms, and to address the aforementioned security challenges.

1.3 Research Problem

This dissertation focuses on proposing and evaluating new defense mechanisms for security purposes for both online social networks and crowdsourcing. We first address the problem of non-adversarial colluding, where a member of the colluding attack does a review on a product while others slightly copy and make some modifications to submit the review as their own work to get paid, and how to detect overlapping colluders with different colluding group sizes. Second, we address the detection of colluding Identity Clone Attacks based on the friend requests received by a user; the goal is to discover the biggest possible number of members involved in the colluding attack. Third, we investigate an advanced collusion strategy in a targeted reconnaissance attack where malicious users can uncover a specific target while having partial knowledge of the network and sending fewer friend requests. Last, we address the problem of colluding targeted reconnaissance attack, where a specific user (target) receives many friend requests coming from different users (colluders and legitimates) and does not want to accept malicious friend requests.

1.4 Research Objectives

The more information sharing among users or/and organizations are secure in online platforms, the better OSN providers will be interesting and gaining users' trust. Our research goals involve addressing existing security concerns and propose robust defense mechanisms to prevent colluding attacks. This dissertation includes new threat models, methodologies, and evaluation results of different defense mechanisms in crowdsourcing and online social networks. We investigate the following four topics:

Detection of Collaborative Malicious users in Crowdsourcing

Crowdsourcing marketplaces or internet crowdsourcing systems, such as Amazon Mechanical Turk (AMT), offer monetary rewards in return for completing certain tasks, and as a result they have become valuable sources of feedback for new products and services. Usually, these tasks are difficult for computers to complete (e.g., sentiment analysis of text, classifying website content, creating a 3-D photo tour, etc.) [GMJM$^+$16], [TNLM15], but this monetary reward has begun to attract malicious users who wish to complete the task with minimal effort through collaboration. A misrepresentation of the true quality of products would be observed in case malicious users present that behavior; colluders are concerned with avoiding the effort normally required to produce a genuine review, rather than with its quality. This new type of non-adversarial colluding behavior in crowdsourcing was proposed by [KCJ14]. [KCJ14] revealed that this type of attack skews many of the metrics normally gathered through reviews, making opinion gathering statistically useless. Additionally, an adversarial variant of this type of attack exists, wherein attackers seek to demote or promote specific products during the review process. Previous works [MLG12], [AIB$^+$12], [KCJ14], [AIB$^+$13] make a strong assumption of fixed clique sizes which fail against attackers that are participating in multiple groups with different sizes, known as overlapping colluding. Therefore, we investigate different malicious crowd sizes which cooperate on different tasks, then we propose a defense mechanism to detect fake reviews and uncover this new type of colluding behavior.

Prevent Identity Clone Attacks Collusion in Online Social Networks

Information sharing between users or organizations has created new threats to OSNs, such as Identity Clone Attacks (ICA). In an identity clone attack [BSBK09, FGE14], malicious users clone other users in a network and impersonate them to infiltrate social circles to gather information from trusting users. Previous research on ICA [JTJ11, KPIM11, HCSS14] predominantly focuses on a single attacker and makes strong assumptions on malicious user behavior. [HCSS14] does not consider the case where multiple attackers could add or modify attributes on the clone. To exacerbate this, an ICA performed by a single attacker is weak, as ICA requires strong prior knowledge of the network in order to be successful; colluding identity clone attacks do not have this weakness and should instead be considered sophisticated attacks. Colluding ICAs take advantage of the profile features of users and exploits the friendship formation among users in OSNs, meaning that detecting the modification of profile features is critical to ensure the security of legitimate users against colluding malicious friend requests and to eliminate malicious clone profiles. Therefore, existing approaches need to be improved. Our objective is to propose a new defense mechanism against colluding identity clone attacks in online social networks.

Attack Strategy for Colluding Targeted Reconnaissance Attack in Online Social Networks

Studying and understanding the way attackers infiltrate a network without being detected is the key to develop security preserving countermeasures. We have seen the growth of various types of attacks in OSNs which can be classified as classic threats (phishing attacks, spammers, etc.), modern threats (clickjacking, inference attack, de-anonymization attack, etc.), and threats targeting children (online predators, risky behaviors, cyberbullying, etc.) which can be damaging to a targeted user [FGE14]. One novel yet important attack in OSNs is the *reconnaissance attack* (RA), wherein an attacker sends friend requests to various users in the network in order to explore and gain information from the network [ND16]. An extension of the RA is the *targeted reconnaissance attack* (TRA), in which the attacker wants to befriend a specific set of target users in order to extract private information [ND16]. The underlying basis of TRA is the Triadic Closure theory, which states that people are more likely to befriend other people with whom they share more mutual friends [BMBR11], and has been shown to apply to OSNs [BMBR11]. But the restriction of topological visibility (default privacy setting for many OSNs such as Facebook is 'friends of friends') make it harder for attackers to reach their target within a minimum number of friends requests, as too many friend requests sent would trigger OSNs defense mechanisms [CSYM15]. In a colluding TRA (CTRA) scenario, colluders share topological knowledge of the network and use sociological theory behind friendship formation to influence their victims, this allows them to easily find a path to a target despite incomplete information of the network topology and while remaining within a minimum number of friend requests sent. Hence, there is a significant need to model and analyze colluding malicious users utilizing TRA in online social networks to evaluate the users' trust and OSNs vulnerability. Our objective is to model an attack to better understand the threat scenario before presenting the defense (Chapter 6).

Defense against Colluding Targeted reconnaissance Attack in Online Social Networks

Friend spam [SCM11, CSYM15] consists of sending multiple friend requests to users in OSNs even though they are not related to that user with the aim to have as many users accept their friend requests. In colluding targeted reconnaissance attack (CTRA), attackers build their own legitimacy by sending friend requests to users that share some similarities (e.g, number of mutual friends, attribute similarities); therefore, making it easier to fool genuine OSN users into accepting their friend requests. Attackers in a colluding ICA can be easily detected because they build new social circles to lure their targets; therefore, discovering one colluder will reveal the others. However, attackers involved in a colluding TRA do not build isolated groups of friends and do not connect with each other; a characteristic that makes it harder for them to be detected by existing defense mechanisms [CSYM15, BBR13]. In this scenario, existing approaches need to be improved because both topology and behavior characteristics need to be included, to the defense mechanism, from each user that sent friend requests to different users in online social networks. Our last objective is to model a threat scenario and propose a new defense mechanism against colluding targeted reconnaissance attack in online social networks.

1.5 Research Contributions

In this dissertation, we investigated the security challenges encountered in crowdsourcing and OSNs. We focused on developing novel solutions that involve (1) detection of non-adversarial overlapping colluding behavior which deceives and misleads crowdsourcers; (2) detection of colluding ICA that may try to befriend the target by sharing information between each other; (3) developing a new colluding attack strategy using RA to measure the OSNs vulnerability because we believe that modeling and exploring the attackers technique is essential for preventing the attack and identifying the countermeasures; (4) proposing a defense mechanism against colluding TRA that will monitor friend requests sent by any users to a specific target to uncover genuine and malicious users.

Detection of Non-Adversarial Overlapping Collusion in Crowdsourcing [KPI+17b]

Unlike the aforementioned works [NWCH14, KCJ14], we address the problem of detecting colluding in crowdsourcing. Specifically, we detect different colluding groups with different clique sizes as well as workers that may collude over many products (overlapping) based on the behavior characteristics of non-adversarial colluders in crowdsourcing. This method calculates (i) the similarity between all the textual reviews on each product given by each worker, (ii) isolates similar reviews on each product, (iii) and classifies workers with high similarity across multiple products as colluders. Afterward, (iv) propose a community detection algorithm to find overlapping colluders in the data.

In this work, we made the following contributions: First, this work addresses and proposes a new methodology to mitigate a novel type of collaborative malicious users involved in product reviews in crowdsourcing platforms. We create an extension to our modified Monge-Elkan (FuzzySim) similarity measure by extracting parts-of-speech (POS) patterns from a text and using lemmatization in order to bring textual differences closer together [Section 3.3.1]. After that, we present our first method to demonstrate that communities (groups) may be well-described by a model that joins small sub-communities (subgroups) to produce a network of collusion from a list of reviews [Section 3.3.2]. We also present a second method to identify the overlapping groups by first finding local sub-communities and then identifying all the communities in the network [Section 3.3.3]. Finally, we discuss the advantage of including both POS and lemmatization to highlight the semantic between words for detecting collusive reviews in crowdsourcing. The simulation results present that our proposed hybrid similarity with POS and lemmatization performs better than the vectorial similarities which were proposed to solve the collusion review in crowdsourcing. In addition, our proposed method to uncover overlapping colluders in crowdsourcing achieves an accuracy of 93% which would be suitable for mitigating this type of collaborative malicious behavior [Section 3.4.2]. This content was published during my Ph.D. study [1].

Preventing Colluding Identity Clone Attacks in Online Social Networks [KPI^{+}17a]

We propose a three-step method to detect ICA colluders in OSNs. Our method matches two user profiles from two different OSNs based on classification techniques that use features extraction on the users' friend request information and friends list. We used a user's attributes profile comparison to rank the probability that multiple users' friend request from different users can be colluders. The first step is to gather profile information of the users' friend request and search through the same or other OSNs for profile that are similar and returns the accounts that are the most similar

[1]© [2017] IEEE. Reprinted, with permission, from [Georges A. Kamhoua, Niki Pissinou, S.S. Iyengar, Jonathan Beltran, Jerry Miller, Charles A. Kamhoua, Laurent L. Njilla, Approach to detect non-adversarial overlapping collusion in crowdsourcing, 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)]

based on the information gathered. The second step verifies the identity of each of the user friend request through the friend list. The final step returns which of those friend requests are colluders. Unlike previous works [JTJ11, KPIM11, HCSS14], our method uses a hybrid-based, string matching similarity algorithm to find profile similarity, and it achieves high accuracy detection on OSNs [CRF03]. We accomplished this by using a recursive matching scheme that includes an internal similarity function for token-based comparisons.

The significant contributions we made in this work are as follows: This work proposes a novel threat model of colluding identity clone attack in online social networks, which allows us to cope with a real-world user's behavior in collaborative malicious friend request sent to a target. Hence, we propose a learning model to uncover the colluding users. The proposed methodology has the advantage of being practical due to the exploitation of users features that are publicly available. To mitigate this type of attack, our propose method is based on three ideas [Section 4.4]. First, we point out how the limitations of the original Monge-Elkan and propose a modified similarity algorithm, called FuzzySim, that will overcome these limitations and outperform other metrics. The significant limitations of the Monge-Elkan method when it comes to document comparison are its asymmetric property and semantic relationships between words [Section 4.5]. To avoid tagging different users that present similar profile information in a real-world scenario, our new methodology employs a friend list similarity algorithm to differentiate them based on the friend name overlapping count, which is calculated by defining the total number of common friends with matching names in the friend's lists between two profiles. Simulation results show that our methodology can mitigate colluders in online social networks even though in the presence of a greater number of colluders. The proposed defense mechanism has a detection average of 80% of malicious users involved in a colluding attack [Section 4.6]. Parts of this subsection have been published during my Ph.D. study [2].

Advanced Colluding Targeted Reconnaissance Attack in Online Social Networks

Reconnaissance attacks (RA) intelligently target OSN users as a means of extracting information for sale or forthcoming cyberattacks. The information targeted can be the one share between a specific user and their friends such as sensitive (health conditions, opinions, employer information, political view, etc.). The aim of targeted reconnaissance attack is to uncover a specific target in the network under some constraints such as partial knowledge of the network. This can be achieved by befriending specific users who may be close to the target (friend of the target, friend-of-friend, friend of the friend-of-friend). Modeling and exploring attackers' techniques used to infiltrate a network without being detected is the key to identify and develop countermeasures to the attack. Restriction on the topology visibility of the network, limited number of friend requests sent, and finding a path to the target are some of the challenges that attackers have to overcome to attain visibility of the target. We address the problem of Colluding Targeted Reconnaissance Attack (CTRA) in OSNs with a minimum number of friend requests sent and incomplete information of the network topology.

Previous works [LSDT16, ND16, LST17] have only considered constraint on the network topology to reach the target, therefore we propose a novel colluding attack scenario for TRA that will allow collaborative malicious users to find a path to a

[2]© [2017] IEEE. Reprinted, with permission, from [Georges A. Kamhoua, Niki Pissinou, S.S. Iyengar, Jonathan Beltran, Charles A. Kamhoua, Brandon L. Hernandez, Laurent L. Njilla, Alex Pissinou Makki, Preventing Colluding Identity Clone Attacks in Online Social Networks, 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)]

specific target with incomplete information of the network topology within a minimum number of friend requests sent [Section 5.4]. Current research utilizes several different approaches including iterative crawlers [ND16], adaptive information benefit maximization [LSDT16], and batch friend requesting [LST17]. These attacks use an 'information benefit' metric based on network topological visibility to help direct the attacker towards the target, but default privacy settings for many OSNs, such as Facebook, are "friends of friends" (i.e., 2 hops away). Machine-learning based metrics do not require topological visibility, they require a substantial amount of training data, which is a serious concern given the scarcity of available OSN data.

Recent security crackdowns by OSNs due to events such as Cambridge Analytica [CGH18] have further complicated access to data, reducing the feasibility of machine-learning. To resolve this issue, we developed an information benefit metric that quantifies a user's homophily, or their tendency to preferentially form friendships with other users sharing some degree of similarity in interests and features, to approximate the closeness of various users in the network [Section 5.5]. Homophily is the user's tendency to preferentially form friendships with other users sharing a certain degree of similarity in interests and features [MSLC01]. It has been found that attributes such as common hobbies/interests, closeness of mutual friends, schools attended, etc. significantly influence a user's decisions regarding friend requests [RBJB14] [Section 5.6]. Finally, we provide an iterative, adaptive attack strategy to gain visibility of a target while sending less than a maximum threshold of friend requests; each accepted friend request updates the network topology, visibility, and dynamic. This strategy updates with each friend acceptance and balances between exploration and exploitation to ultimately gain visibility of the target [Section 5.7]. The experimental results obtained in this work show a notable difference in performance, which indicates that our proposed strategy surpasses other strategies. In addition, we investigate the difference between a colluding and a single attack in TRA: our single TRA was able to reach 86.5% of success on the attack trials (2000) within 200 friend requests to attain the target, while alternative strategies were at most at 75%. In contrast, 94% of the attack trials (2000) attain visibility of the target within 40 friend requests sent on CTRA, while 100% have attained the target within 60 friend requests sent. Also, our CTRA exceeds in performance in terms of magnitude compared to alternative strategies, which were at most 91% within 200 friend requests sent to reach the target [Section 5.8]. Some parts of this content have been submitted for publication [KML$^+$19].

Detection of Colluding Targeted Reconnaissance Attacks in Online Social Networks

To address the problem of detecting collusion in TRA, we monitor the friend requests sent by any users to specific targets that we classify as malicious or normal. The goal is to efficiently identify colluding malicious friend requests sent by users to specific targets while avoiding detecting normal users as colluders. Afterward, we address the problem mentioned in Chapter 5 as a security challenge. To be resilient to the collusion behavior observed in OSNs, we formulated the colluding malicious friend requests as ego-centered networks and overlapping community detection problem. Current research [CSYM15, ITI$^+$17a, VLC$^+$17] has the following specific characteristics to assign a node in a community: 1) it views the network as a whole (i.e., consider all the nodes in the network to define a quality function) [CSYM15]; 2) it does not consider the particularity of each node (i.e., community formed by each node in the graph); and 3) either it considers the user topology [ITI$^+$16], [ITI$^+$17a], [ITI$^+$17b] or behavior [ESKV17], [JCB$^+$16], and [VLC$^+$17], but not both.

Our main contributions can be described as follows: First, we design a colluding detection mechanism for targeted reconnaissance attack under an Ego-Centered Network model, that performs classification between legitimate or malicious friend requests sent to a specific target [Section 6.4] and [Section 6.5]. Then, we propose a defense mechanism against colluding in targeted reconnaissance attacks based on a node-centric approach, wherein the target does not know the entire topology of the network, just the friends lists of each user that sends a friend request. As a result we have designed a defense mechanism that builds a community around each node [Section 6.6]. After that, we discuss the advantage of using an Ego-Centered Network model that includes both graph-based (topology) and behavior-based (communication) characteristics in OSNs. This model would allow us to cope with a real-world user's behavior rather than just considered one of the characteristics [Section 6.7]. The simulation results show that our proposed defense mechanism against colluding targeted reconnaissance attack presents significantly better performance than the others. For a threshold of 0.7 on different datasets, our proposed approach shows a true positive rate of 0.615%, 0.77% and 0.7% for detection of malicious users involve in a colluding targeted reconnaissance attack, while other approaches were at 0.42%, 0.55% and 0.53% maximum respectively. The superior results obtained by our model obtained emphasize the inclusion of both graph structures and user behaviors for a better defense mechanism [Section 6.7]. Some contents of this section have been submitted for publication [KZP$^+$19].

1.6 Dissertation Outline

The following is a concise outline of this dissertation: Chapter 1 presents a well-detailed introduction to the research in this dissertation; it describes the dissertation background, the overall contributions, and the thesis structure. In Chapter 2, we present a comprehensive overview of the security and privacy challenges in OSNs and crowdsourcing. Chapter 3 provides key concepts and gives a clear view of the relevant colluding behavior used in crowdsourcing. The main objective in this chapter is to propose a method to uncover overlapping colluders that performed task reviews on products in crowdsourcing (Amazon Mechanical Turk). Chapter 4 analyzes the characteristics of different threats that OSNs users can be exposed to and elaborates on our proposed methodology to detect colluding ICA. Chapter 5 will cover the OSNs vulnerability to a new type of infiltration strategy. Inspired by the ICA that involves collusion, we propose a new variation to the RA called colluding Targeted Reconnaissance Attacks. Later, Chapter 6 will elaborate on two essential steps to detect colluding malicious users in reconnaissance attack. It exploits both graph-based (topology) and behavior-based (communication) in OSNs and uncovers malicious friend requests from colluders. The methodology used to combat colluding reconnaissance attacks monitors friend requests sent by any user to specific targets to classify as malicious or normal users. The aim is to efficiently identify colluding malicious friend requests sent by users to specific targets while avoiding detecting normal users as colluders. Finally, Chapter 7 will provide insight into solving the proposed research questions, provides a conclusion and recommendations for further research on this dissertation.

CHAPTER 2

RELATED WORK

Currently, a large part of the literature exists about threats and solutions in online social networks and crowdsourcing platforms. In this chapter, we would emphasize on relevant literature related to the techniques evaluated in this dissertation. It also points out a clear view of related research used for addressing the problem of threats in those platforms. Therefore, Section 2.1 presents the research efforts on collusion in distributed and centralized networks. Section 2.2 provides the current research efforts on fraudulent review detection by malicious users in different systems. Section 2.3 gives the current approaches to detect colluding threats in online social networks. Section 2.4 presents research on the detection of identity clone attack. Section 2.5 presents the current works of exploiting OSNs vulnerabilities to collect precise/personal information of targeted users. It draws a clear view of the relevance between research used on a single attacker and colluding users for infiltration purpose. In addition, it presents the existing works on defense mechanism against colluding targeted reconnaissance attack in OSNs. Our main objective in this chapter is to analyze existing approaches based on different criteria to emphasize on which parts need more exploration to be strengthen against malicious users.

2.1 Collusion in Cyberspace

Collusion Attacks happen when two or more users collaborate to take advantage of breaches in trust. This type of attack will favor every participant involves in the malicious collaboration to maximize their benefit more than each colluder would achieve individually. The following work discussed this spiteful collaboration amongst users in real-world scenarios.

Lian et al. [LZY$^+$07] analized and proposed a defense mechanism against spiteful collaboration among users in peer-to-peer (P2P) file sharing systems. The behavior pattern they mitigated amongst members of the colluding attack is the attribution of unfair positive feedback to increase their reputation in the system. Therefore, colluding members would have the advantage to freely change their behavior compare to legitimate users. Their approach includes different indicators that are based on each user behavior (e.g., user logs). However, the proposed approach suffered from a large number of false positive due to legitimate users that present similar behavior as colluding members. Daubert et al. [DGMF15] analyze the colluding internal attacks on pub-sub anonymization systems. They determined that the number of colluders in the network does not affect the outcome of detection. Li et al. [LSS13] proposed an approach called SocialTrust to detect colluding users in OSNs for systems that implement reputation systems. Colluder users are socially close to each other and rate each other with high scores. SocialTrust modifies the weight of ratings based on the social distance and interest relationship between peers, which increases the ability of the reputation system to fight against colluding. They also claim that their mechanism can be used in any reputation system for P2P networks, but they only tested this in the eBay environment with the assumption that all users in the network are legitimate and rational nodes. Niu et al. [NWCH14] characterized collusive manipulation on ratings in Amazon. They modeled normal behavior and determined that colluders are disconnected from other users. However, the model is prone to overfitting, and comparative experiments were not conducted for their approach. They also did not provide the way they collected their data. Chen et al. [CLSQ15] introduced a trust management system called SocialLink that uses OSNs. They prevented colluding by decreasing link weights when given faulty files. Their system is based on the assumption that users will not want to damage real-life reputations. Parno et al. [PPG05] identified that a clone attack (or node replication attacks) in sensor networks is equivalent to ICA on OSNs. In node replication attacks, the malicious nodes apprehend some sensors, replicates them, and deploys them again in the networks. The purpose here is to perform internal attacks into the networks through these compromised sensors. However, the detection process in sensor networks is different from the one of ICA in OSNs for two reasons. First, sensors in the networks aim to communicate with each other, whereas the users in OSNs do not need to communicate or collaborate with each other all the time. Second, for OSN sites, the primary stride of ICA is to uncover similar profiles. Similarity uncovering in sensor networks may be trivial because the cloned nodes in sensor networks are the same as the victim nodes. Another major contrast illustrated by Jin et al. [JLTJ12] is the difference between Sybil and ICA. The similarity on both comes from the fact that the attackers created profiles to perform malicious activities. However, in Sybil attacks, the attacker creates many unique profiles to begin an attack against OSN users. While in ICA, the attacker clones the victims (friends of the target) in an attempt to deceive the target to trust the clone profiles to gain private information. Also in ICA, an attacker must have some previous knowledge about the target identity.

2.2 Users Collusion in Review Systems

In this particular section, three related areas that are similar to our proposed approach will be analyzed. We also want to emphasize on the approaches of detecting colluding behavior of users in those platforms, which are useful approaches for detection. Therefore, 1) discusses on the current research of detecting fraudulent reviews on online rating system; 2) gives the existing work on fraudulent feedback on online reputation system; 3) provides the existing approaches of fraudulent review detection in crowdsourcing platforms.

2.2.1 Detection of fake reviewers in Online Rating Systems

Mukherjee et al. [MLG12] proposed a method to detect fake reviewer groups by analyzing textual feedback given on products in the Amazon online market. They used the FIM [AS$^+$94] algorithm for the detection of candidate groups and use height indicators to identify groups of colluders. Lim et al. [LNJ$^+$10] studied the detection of review spammers. They gave each of the reviews a degree of spam value and based on those values, they determined the most suspicious reviews and used predefined types of behavior abnormalities to find colluders. Allahbakhsh et al. [AIB$^+$12] studied the impact of the collusion attacks in online rating systems. They created a model to represent the relationship between reviewers and products, called biclique, and also used a method based on six indicators to distinguish the difference between honest and collusive groups. In [AI15], Allahbakhsh et al. used iterative techniques to identify groups of colluders in the log of online rating systems in Amazon. In [AIB$^+$13], Allahbakhsh et al. proposed a collusion detection algorithm based on indicators to detect possible collusive groups among reviewers.

Unlike previous research, our work focus on the of detection of overlapping collusion campaigns launched by malicious users. Our approach takes advantage of the real-world behavior that some colluders in crowdsourcing platform can present. For instance, a malicious user can collaborate with different clique size of users in different products. Therefore, the research mentioned above will not be able to detect this type of behavior.

2.2.2 Manipulation of Online Reputation Systems

Kamvar et al. [KSGM03] proposed a well-known algorithm called EigenTrust to detect groups of colluders by using the feedback of trusted nodes. [LZY$^+$07] established that EigenTrust does not perform well enough against collusive attackers. The authors of [LYS08], [YSKY09] showed there exist substantial interactions between malicious colluding users. Daubert et al. [DGMF15] analyzed the colluding internal attacks on pub-sub anonymization systems. They determined that the number of colluders does not affect the outcome. Niu et al. [NWCH14] studied and proposed an approach against collusive manipulation on ratings in Amazon. They modeled normal behavior and determined that colluders are disconnected from other users. They addressed the false positives in their algorithm but did not conduct a comparative experiment for their approach. Moreover, they did not explain the rules for their filter in order to make their web crawler.

Our work differs through its significance on the relationship among different sets of reviewers in the context users behavior in crowdsourcing system. Malicious users may or may not want to upgrade or downgrade the products they are writing reviews for; However, they are just interested of receiving monetary payment for tasks that were not genuinely completed (not spending time to review different tasks). For instance, collaborative malicious users can copy reviews between them and proceed for little modification on them before submitting as their own.

2.2.3 Non-Adversarial Users in Crowdsourcing Systems

Crowdsourcing systems such as Amazon Mechanical Turk (AMT) have attracted the interest of many researchers; however, they have not addressed the possible misuse of crowdsourcing systems. Mason et al. [MW09] demonstrated workers who wanted to increase their remuneration corresponds exactly to the time or energy spent on finishing a task. Workers that want to have more money for the least amount of time spent working may try to act maliciously. Wang et al. [WWZZ14] detected malicious accounts performing ”crowdturfing”, or fake crowdsourced-grassroots efforts on Twitter and Weibo. KhudaBukhsh et al. [KCJ14] proposed an unsupervised collusion detection algorithm to classify colluding groups in crowdsourcing. Their algorithm analyzed rating scores cast on products in a large e-commerce organization. They detected collusion by finding strong inter-rater dependence across products ratings, precisely as these diverge from the mean ratings. They use Kullback-Leibner to measure the corruption of the distribution. This measure is undefined if $Q(i)$ is zero. It is not symmetric because the divergence from P to Q is not the same as the one from Q to P. Our model differs from their model because we are using textual feedback instead of just a rating. We also deal with products where more than 50% of the reviewers can be malicious, analyzing a different type of colluder (overlapping). Furthermore, the model proposed by KhudaBukhsh et al. is vulnerable to simple false positive cases. For example, if a group of multiple users holds the same opinion, their method would classify them as colluders. If there are more colluders than real reviewers, it will classify the colluder’s behavior as normal.

2.3 Collusion in Online Social Networks

In this specific section, we will investigate two related areas similar to our work. 1) We discuss attempts made by others works to detect Sybil in online social networks; 2) we complete by discussing some existing work on communities detection mechanisms on colluding attackers, which are more related to the problem we want to solve, and we explain how our work presents aspects to perform better in colluding detection.

2.3.1 Sybil detection mechanisms

Fire et al. [FGE14] proposed a general literature on malicious activities that OSN users are exposed. Malicious activities such as malware, spammers, etc., have used social engineering attack strategies to steal information of legitimate users and furthermore compromise their account to launch malicious campaigns. Egele et al. [ESKV17] analyzed and proposed a compromised accounts detection framework, called COMPA in OSNs. Their framework is based on a behavioral model using the pairwise similarity method to identify compromised accounts on Facebook and Twitter. However, the definition of normal user behavior is complex in the real world of OSNs; a small deviation of normal user behavior can cause a problem for their model that relies on pairwise similarity. Moreover, an attacker can evade their detection by sending messages as a genuine user.

Those malicious accounts, known as Sybils, created by attacker(s) are injected into the system to compromise the security and privacy of its users. Therefore, they make the system vulnerable and more attractive as target for malicious users. To combat Sybil attacks in OSNs, research has leveraged the social graph structure-based approaches. These techniques use the graph-theoretic difference between legitimate and Sybil users. Generally, an OSN represents a graph with nodes and edges defining users and friendships among them, respectively. Given the key assumption that Sybil users can befriend small legitimate users (attack edges), these techniques partition the social graph into two distinct areas (i.e., legitimate and Sybil). As a result of this underlying assumption, research such as [GFM14, ACE$^+$13, JWG17, WZG17] has been proposed to detect Sybil attacks. However, Yang et al. [YWW$^+$14] have shown Sybil attacks can a large number of attack edges. Therefore, Sybil attacks can easily evade detection based on social graph. Another approach to improve Sybil attacks detection is to include more information on these schemes such as acceptance/rejection of friend requests [BLS+16, EY17, ZXL18, YXY+15]. Even though with these new schemes, a small behavior change (i.e., reconnaissance attack [PSP15]) from Sybils would result in large amount of false positive detection. Different from previous works, we aim to include both social graph structure-based and activities/interactions among users for better detection scheme.

2.3.2 Community Detection

The community detection has been used for Sybil attacks. Viswanath et al. [VPGM11] propose a local community detection approach around a labeled legitimate user. However, the method relies on manually labeled legitimate users for Sybil detection. Cao et al. [CYYP14] developed a system, called SynchroTrap, that detects malicious accounts that have synchronized activity patterns (i.e., similar actions performed during the same period of time). Also, Jiang et al. [JCB+16] proposed a method, called CatchSync, to automatically spot anomalous, suspicious behavior only from their connectivity pattern that is based on synchronized activity on Twitter. Vo et al. [VLC+17] first proposed an algorithm, called Attractor+, to detect malicious retweeter groups, and then introduced group-based features to catch synchronized and coordinated behavior. However, malicious accounts can modify their activity patterns such as not acting synchronously to avoid detection. Hence, Stringhini et al. [SMJ+15] proposed a system, called EvilCohort, that identifies accounts which are accessed by a regular group of malicious users. Their method required a mapping between IP addresses and user accounts to build a bipartite graph (i.e., set of online user accounts and set of IP addresses) to perform clustering; it will help the system to unveil common set IP addresses that are accessed by a common set account communities. Nevertheless, malicious users can evade this type of detection approach by adjusting the IP mapping strategy.

The previous research mentioned above differs from our work. None of them rely on sending friend requests to other users in the network. In addition, they suffer from intelligent malicious users behavior who may forge legitimate community structure as legitimate users. Therefore, this would make mechanisms based on community analysis ineffective for detection.

2.4 Detection of Identity Clone Attack in Online Social Networks

We investigate existing work associated with methods for identity clone attack detection across one or more social network(s). Most OSNs have different privacy settings to enable users to protect their personal data. Despite this, most users still maintain the default privacy settings that allow the data to be exposed. Previous works have created methods for detecting these fake profiles that use this publicly available data.

Jin et al. [JTJ11] proposed a detection framework that uses attributes of the user and friend network similarity as a detection method. Their framework was designed for detection purposes only, so it will fail to defend users against ICA. This method lacks the process to determine equivalency such as token similarity or character-based similarity, which leads to misclassification of similar profiles. Their framework also assumes that the victim and the clone exist on the same OSN, so they cannot detect cross-site profile cloning. Kontaxis et al. [KPIM11] gave a methodology for detecting social profile cloning. They use exact string matching for detection in the same OSN. Their work does not consider attacks involving multiple OSNs. Shan et al. [SCL$^+$13] improved the ICA model and developed a detection method called CloneSpotter, which makes decisions about whether a friend request is a clone. This is based on user attributes and real connection by using IP addresses. The main drawback of their detection process is that it is reliant on IP tracking. Therefore, a true attacker can use anonymous proxy servers, a common technique to mask IP address, or Virtual Private Network to fool the CloneSpotter. He et al. [HCSS14] proposed an approach based on three techniques to determine an ICA: issuing a challenge to verify the friend request identity, using login as an identifier to be connected to many social networks, and counting the number of mutual friends. However, we do not stop at detecting an ICA, we propose a method to identify ICA colluders.

Colluder ICA may change some profile information to avoid detection for the exact profile matching framework. We discovered a fundamental limitation in cross-matching profile methods. Our work is unique to the previous work in that we detect a group of clone profiles working together (colluding). These colluders aim to infiltrate the target's social circles in social networks with high accuracy. We also detect whether the profiles in question are cloned or genuine.

2.5 Infiltration Process and Defense in Targeted Reconnaissance Attack

In this section, three main areas related to our work will be investigated. 1) We will point out the threats posed by the reconnaissance attack in online social networks and analyze the current work on reconnaissance attack strategy; 2) we focus on attack graph strategy, which are very useful approaches for an attack in a network; 3) we present the existing work on defense against reconnaissance attack.

2.5.1 Description of Reconnaissance Attack

Online social networks allow their users to create relationships and share information without knowing the security and privacy risks they are exposing. Sometimes users reveal personal information or share posts that may contain sensitive information harmful for users or organizations. For example, the Israeli military canceled an operation after a soldier posted details on Facebook [MAC$^+$11]. Blige et al. [BSBK09] showed that most people on OSNs are not careful while receiving a friend request. The authors conducted an experiment to test how users accepted friend requests and explained that the acceptance rate was over 60% for forged profiles (already in their friend list) and 20% for fake profiles. Authors of [SE13, SSM$^+$16] showed how RA could be damaging and crucial; attackers collect extracted information via OSNs on target organization resources, users, and relationships with other people that can be used to reach the target.

Preliminary research in Reconnaissance Attacks (RA) include the use of social bots [BMBR11, FVD$^+$16], which are machine operated fake profiles in OSNs that target real users to friend and infiltrate organizations to later perform malicious activities. Fire et al. [FP16] designed a targeted crawler capable of infiltrating organizations. They combined the homophily with machine-learning to prioritize which users to explore. They were able to uncover hidden departments and key roles of workers among many discoveries in each organization. Targeted Reconnaissance Attacks (TRAs) have been built utilizing targeted crawlers [EFKE12, EFKE13, PSP15] to first obtain prior knowledge of network topology. However, recent API restrictions by OSNs, such as Facebook due to Cambridge Analytica [CGH18], put in place to increase user privacy have made crawlers and scrapers much less feasible.

There is currently less defensive research being done on Reconnaissance Attacks. Methods used for Sybil defenses rely on topological partitions [BBR13]. However, since Reconnaissance Attackers do not rely on a network of Sybil users, these methods are not applicable. Paradise et al. [PSP^{+}17] used Honeypots to detect Reconnaissance Attacks. While their method of deploying these Honeypots was largely successful, their detection of malicious behavior was inconclusive.

The four works that are most relevant to our paper are in [ND16, LSDT16, LST17, PSP15]. However, these works can be detected by monitoring the networks using social rejection in OSNs [CSYM15]. None of these works use the sociological theory relationship behind friendship formation in OSNs to better implement the infiltration strategy. Therefore, these works suffer for a high rate of their friend requests rejected. We observe that we are the first to investigate TRA by using homophily based similarity metrics with incomplete information of the network.

Another approach proposed by Gong et al. [GL16, GL18] exploits both users' social friends and behaviors information for inferring users attributes. They created a framework, called social-behavior-attribute network (SBA), in which social structures, user behaviors and user attributes have been combined. However, user privacy can be compromised because of the combination of structure and external source of information. Liu et al. [LL16] proposed an advanced collusion attack that violates a victim user's friendship privacy setting on a friend search engine. They were able to achieve it by a well-designed synchronizing query sequence started by many spiteful users. However, this work is limited on a fixed number of friends to be displayed by the victim over malicious users; they did not consider different values of the number of friends to be displayed or apply their strategy on large real-world scale social networks. Liu et al. [LL18] proposed an advanced collusion attack strategy where many malicious users with limited initial knowledge (i.e., only on the victim) can invade the defense and compromise victim's user privacy setting on the friend search engine. However, this work is only related to the network topology.

None of these works use the sociological theory relationship behind friendship formation in OSNs to better implement the infiltration strategy. Therefore, these works suffer from a high rate of friend requests rejected [CSYM15].

2.5.2 Vulnerabilities Analysis using Attack Graphs

Attack graphs represent possible ways a malicious user can utilize to reach/deceive a target in a system. This type of attack is feasible by the exploitation of potential vulnerabilities existing inside a system. Du et al. [DY11] investigated cooperation between malicious users by analyzing their social network's activities. They defined a new attack to evaluate collaborative patterns between attack sources. However, their approach is limited to uncover cooperative adversaries that perform related patterns. Munoz-González et al. [MGSPL17] provides a comprehension on the Bayesian attack graph; which evaluates and model the ability/probability an adversary can penetrate a system based on its vulnerabilities. While in [MGSPL17], Munoz-González et al. model Bayesian network for attack graph and propose efficient inference techniques for their different analysis (static/dynamic). In the case of a cyber-attack against a node in the network, those techniques provide the probability that an attacker can compromise each node given the nodes that have been already compromised. Recently, Chen et al. [CNK$^+$17] design and evaluate two novel attack graphs against the network level. They focus on adversarial machine learning, precisely graph-based clustering techniques, that have not been discussed in the attack graph. The authors demonstrate that malicious users who carry out these attacks can efficiently circumvent graph-based clustering techniques with limited knowledge and low cost.

However, this type of attack differs from the problem we intend to solve; which is to befriend a number of users during the infiltration process to reach a target.

This has to be done within a minimum of friend requests, limited knowledge of the network topology, and user to whom attacker send friend request can accept, reject, or ignore it.

2.5.3 Mitigation in Reconnaissance Attack

Paradise et al. [PPS14] used honeypots to detect malicious users in RA that connected to employees in a targeted organization. Their method was based on wisely selecting organization member profiles, monitoring, and investigating their profile activity. In contrast, Paradise et al. [PSP$^+$17] proposed a framework for management of social network using honeypots to detect RA in order to avoid the obligation of monitoring profiles activity of real employees. Li et al. [LSDT16] quantified an intuitive metric, called Reconnaissance Resistance Score, which is the mean number of friend requests to get one unit of the information benefit. However, the detection of RA was not successful because of its dependency on the network topology. Previous works do not study the case of colluding behavior.

2.6 Summary

In this chapter, we have analyzed the existing main approaches to protect legitimate users in online social networks and crowdsourcing platforms against colluding attacks. The previous works show the importance of our work and give a clear view of the problem we want to solve here. In the following chapters, we will present our results and provide some important future directions on this topic.

CHAPTER 3

DETECTION OF COLLUSION IN CROWDSOURCING

In this chapter, we address the problem of detecting deceptive and malicious behavior seen in product reviews in crowdsourcing platform. Crowdsourcing services have become one of the most common ways organizations can gather ideas for new products and services from large crowds of consumers by offering monetary rewards depending on the tasks. However, this financial reward has begun to attract malicious crowds of users who wish to complete the task with minimal effort through collaboration. For instance, a task based on reviews of a product can be degraded when malicious users copy each other with minimal edits of the review, giving a misrepresentation of the true quality of the product. More specifically, we investigate the case where different malicious crowd sizes cooperate on various tasks, known as overlapping groups. Such sophisticated and hard to detect malicious crowds provide unfair evaluations and misleading results to the crowdsourcers. The outline of this chapter is organized as follow: A general overview of the collusion attacks encounter in review systems with a particular emphasis on crowdsourcing system is presented in Section 3.1. Section 3.2 presents the problem statement. The proposed method will be introduced in Section 3.3. Then in Section 3.4, we present our simulations, results, and evaluations of our proposed approach using the Ott dataset and compares it to the similarity measures mentioned before. Finally, Section 3.5 gives a summary of this chapter.

3.1 Introduction

Crowdsourcing marketplaces or internet crowdsourcing systems have allowed a way for completing tasks which are difficult for computers by exploiting the power of available human workers from across the world to perform these tasks [TNLM15], [GMJM$^+$16]. Such problems include classifying images or videos [VAD08], matching data records that fit the same entity [WKFF12], sentiment analysis of text [SOJN08], creating a 3-D photo tour [SSS06], and creative tasks such as translating books, classifying website content, or collecting data about the real world. One example of innovation through crowdsourcing was the Toyota Logo Contest in 1936 to redesign its logo. The company accepted 27,000 different entries before selecting their winning logo that is still used today [Lyn10].

One of the major reasons why crowdsourcing has become so widespread over the years is the convenience of internet platforms such as Amazon Mechanical Turk (AMT), UpWork and Samasource [GMJM$^+$16]. They allow for large scale distribution of tasks to human workers while automating the payment process. Even though crowdsourcing makes the collaboration of large-scaled workers easier, a key feature of crowdsourcing comes from using other people's feedback to analyze the quality of tasks, which provides credibility of the feedback due to large, diverse sources of opinions. However, since workers can be recruited from any part of the internet, this could lead to selective manipulation of the data. How can we ensure the answers are diverse and genuine? A collaborative malicious group of workers, known as colluders, can work together to fool the crowdsourcers with misleading feedback on the current task. Collusion detection has been studied in many works such as online rating systems [MLG12], [LNJ$^+$10], [AIB$^+$12], [KCJ14], [AI15], [AIB$^+$13] and online reputation systems [KSGM03], [LZY$^+$07], [LYS08], [YSKY09], [SLSL16].

Identifying groups of colluders usually require looking for cliques. This method has been widely studied in recent literature [MLG12], [AIB$^+$12], [KCJ14], [AIB$^+$13]. Nevertheless, some significant challenges remain such as detecting different groups of colluders who may have different clique sizes. Trushkowsky et al. [TKFS13] showed that workers would collude with each other in order to provide the same answer for a given task. KhudaBukhsh et al. [KCJ14] proposed a new type of colluding behavior in crowdsourcing where colluders copy product ratings between each other and make minor changes to the original one to avoid being caught (non-adversarial). The goal of this type of behavior is to receive monetary payment for a task that was genuinely completed. In this attack, workers who perform the task are known as leaders while the workers who copy are known as followers. KhudaBukhsh et al. [KCJ14] revealed that this type of attack creates side effects on many statistical metrics such as the mean, median, and the variance. This makes the opinion gathering statistically useless.

Unlike the work mentioned above, in this chapter, we address the problem of detecting colluding in crowdsourcing. Specifically, we detect different colluding groups with different clique sizes as well as workers that may collude over many products (overlapping) based on the behavior characteristics of non-adversarial colluders in crowdsourcing. The proposed method calculates the similarity between all the textual reviews on each product given by each worker, isolates similar reviews on each product, and classifies workers with high similarity across multiple products as colluders. Afterward, we propose a community detection algorithm to find overlapping colluders in the data.

Furthermore, there is no restriction for one malicious worker to belong to more than one colluding group(s). Therefore, it is possible for some workers to collude on some occasions and not on others, hence different sizes of colluding groups will be involved. Our results show that our method is strong enough to overcome one of the significant challenges when a group of workers is trying to take over a product. For example, when the percentage of colluders is higher (50%) than the genuine participants reviewing the product.

3.2 Problem Statement

Given how workers in crowdsourcing systems can create side effects on many statistical metrics and misleading feedback on opinion gathering, preventing malicious behavior in such system is predominantly important. Therefore, we plan to answer the following question:

Can community detection methods identify non-adversarial overlapping colluders in crowdsourcing?

Our hypothesis was that we could detect non-adversarial overlapping colluders by using a method based on community detection algorithms and semantic similarity measure. In non-adversarial colluding, we aim to uncover malicious workers in different group sizes that may collude over many different products (overlapping). We include the Louvain method for greedy modularity optimization which best suits for large networks. The success of Louvain comes from its bottom-up approach to form communities in a network.

Even though we are exploiting an algorithm based on community detection that would also include the Louvain method to mitigate collusion in crowdsourcing, proposing a new defense mechanism is still challenging. Colluding users can evade the defense mechanism by not just merely copying other reviews of malicious groups, but by actually trying to modify reviews based on semantics. Therefore, document comparison function between words is not seen sufficient. Colluders will spend a little more time to produce reviews but may be sure to collect the pay as each had done the work genuinely. We call this strategy used by colluders non-adversarial overlapping.

3.3 Proposed Method

In this section, we will present our similarity measure, our overlapping detection methodology, and the overlapping collusion detection algorithm.

3.3.1 Similarity Measure

The textual similarity measure is common in most natural language processing problems. Evaluating the similarity between two text documents amounts to saying that to compute the similarity of their words. Two words are considered similar if and only if they have the same meaning or some relation that can be defined between them such as synonym/antonym. The similarity measures are generally used in information retrieval to find the most suitable match set of relevant documents for an input text. Monge et al. [ME+96] proposed an effective and simple recursive matching scheme to compare the similarity between two long strings, A and B. First, A and B are broken into several tokens $A = a_1, a_2, ..., a_i$ and $B = b_1, b_2, ..., b_j$ with $i = \|A\|$ and $j = \|B\|$, respectively. The similarity between A and B is defined as

$$Sim_{MongeElkan}(A, B) = \frac{1}{\|A\|} \sum_{i=1}^{\|A\|} max\{sim'(a_i, b_j)\}_{j=1}^{\|B\|} \tag{3.1}$$

where sim' is an internal similarity function able to measure the similarity between two individual tokens.

As the internal similarity function, we use Jaro-Winkler distance [Jar95] which is based on the number of common characters, m, between two strings and the number of transpositions, t, as follows:

$$Sim_{Jaro\_Winkler}(a,b)\begin{cases} sim_{jaro}(a,b) & if\ sim_{jaro}(a,b) < b_t \\ sim_{jaro}(a,b) + \frac{l}{10}(1 - sim_{jaro}(a,b)) & otherwise \end{cases} \tag{3.2}$$

with

$$Sim_{Jaro\_Winkler}(a,b)\begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}(\frac{m}{\|a\|} + \frac{m}{\|b\|} + \frac{m-t}{m}) & otherwise \end{cases} \tag{3.3}$$

where l is the length of the common prefixes at the start of the string up to a maximum of 4 characters, and b_t is the boost threshold (0.7) in the Winkler's implementation [Jar95]. Two characters, a and b, are considered matching or common only if they are in the sliding window size and not farther than $\lfloor\frac{max(\|a\|,\|b\|)}{2}\rfloor - 1$.

Because existing similarity functions have limitations, to address these problems, we propose a modified similarity metric called FuzzySim that will overcome these limitations and outperform other metrics.

$$FuzzySim_{MongeElkan}(A,B) = \frac{1}{max(\|A\|,\|B\|)}\sum_{i=1}^{\|A\|}\bigcap_{Average\ \theta}\{sim'(a_i,b_j)\}\|_{j=1}^{\|B\|} \tag{3.4}$$

The major limitation of the Monge-Elkan method when it comes to document comparison is its asymmetric property. Table 3.1 and Table 3.2 lists five examples along with the percentage of similarity given by each comparison method. Consider the examples in Table 3.1 where different document comparison functions are presented. As you can see, FuzzySim outperforms Fuzzy Jaccard. When comparing 4(a) with 5(a), our method is more consistent than the normal Monge-Elkan which will lead to better document comparison.

Strings	Fuzzy Sim	Fuzzy Jaccard	Monge-Elkan / Jaro-winkler
1. “Martha/marhta”	0.822	0.00	0.822
2. “Brandon hernandez / hernandez brandon”	0.962	0.867	0.952
3. “Lenovo inc/lenovo corp”	0.500	0.333	0.750
4. “aaaa / aaaa xaaa yaaa”	0.890	0.333	1.000
5. ”aaaa xaaa yaaa / aaaa”	0.890	0.333	0.890

Table 3.1: Comparing different similarity measures (Comparison functions)

Strings	Jaro	Jaro-Winkler	Levenshtein
1. “Martha/marhta”	0.822	0.822	0.500
2. “Brandon hernandez / hernandez brandon”	0.700	0.790	0.250
3. “Lenovo inc/lenovo corp”	0.842	0.905	0.636
4. “aaaa / aaaa xaaa yaaa”	0.762	0.857	0.285
5. ”aaaa xaaa yaaa / aaaa”	0.762	0.857	0.285

Table 3.2: Comparing different similarity measures (Internal similarities)

Our comparison function differs from Monge-Elkan by the use of a threshold, θ, and makes the function symmetric by dividing by the maximum length of the strings being compared. The threshold, θ, allows for making strict comparisons, i.e., if θ is set to a high percentage, it means that when two tokens are compared in sim' they must be at least θ equivalent to be included in the overall score. This leads to better overall document comparison as low scoring comparison, such as comparing nouns/adjectives with prepositions, are ignored as long as θ is set to a higher percentage than an average comparison with a preposition.

Based on these tables above we see that our method outperformed the existing approaches. However, all these methods hold the same limitation when trying to semantically compare two documents. It does not take into account the semantic relationships between words, so text with similar vocabulary are not seen as similar to a document comparison function. Various ways of improving document comparisons have been proposed, such as POS tagging. According to [MCS+06], there are some approaches to measure semantic relatedness and many of them are based on WordNet. WordNet is a large lexical English database where nouns, verbs, adjectives, and adverbs are arranged into groups of words that in the context of information retrieval are considered semantically equivalent, known as syssets. These groups of words are connected through relations into a larger network which can be used in computational linguistics problems.

Michalcea et al. [MCS+06] proposed a method to improve the word-to-word similarity metric inside WordNet to compare entire texts and showed their method outperformed the traditional vectorial methods. They combined the maximum value from six well-known knowledge-based metrics of word-to-word similarity based on distance in WordNet. Word specificity was defined by the inverse document frequency metric, *idf*, which is a good indicator of the semantic similarity of two inputs texts T_1 and T_2.

Based on the behavior of non-adversarial colluders described by [KCJ14], we assume that colluders are susceptible to making small alterations to the original text review. Methods such as replacing words with their synonyms/antonyms all while keeping the overall review text identical. Note that by colluders we mean a group of malicious users' IDs. We emphasize on the actual users behind the users' IDs could be a single user with multiple user IDs, various users, or a combination of both. We assume that each user ID corresponds to a user.

Algorithm 1 is used to evaluate the pairwise similarity between reviews in each set of products. Reviews are textual documents that can be a mixture of many topics. To find possible overlapping colluders, we should compute the similarity on reviews casted by users on each of the products, then check to find semantically similar reviews. The algorithm describes how to compute the pairwise similarity

Algorithm 1: Detection of possible overlapping collusive groups

```
Input  : $RP$ is the multisets of reviews casted by users for each
         product.
Output: $N$ is the set of all possible overlapping colluders on different
         product, initially empty
       : $RP_{ij}$ Reviews casted by user $i$ on product $j$.
       : $TR_j$ is a list of connections between two users on product $j$.
1  Initialize N as a $m \times m$ matrix, $m = \|\text{Reviewers}\|$
2  for $all\ reviews\ R_{i,j}\ in\ dataset$ do
3      Remove stopwords
4      Extract POSs
5  end
6  for $\forall\ RP_j \in RP$ do
7      for $i \leftarrow 1\ to\ \|RP_j - 1\|$ do
8          for $k \leftarrow i + 1\ to\ \|RP_j\|$ do
9              $Similarity \leftarrow FuzzySim(RP_{ij}, RP_{kj})$
10             if $Similarity \geq \delta$ then
11                 $TR_j \leftarrow (i, k)$
12             end
13         end
14     end
15 end
16 for $m \leftarrow 1\ to\ \|TR\| - 1$ do
17     for $n \leftarrow m + 1\ to\ \|TR\|$ do
18         for $\forall\ t \in TR_m$ do
19             for $\forall\ s \in TR_n$ do
20                 if $TR_{tm} == TR_{sn}$ then
21                     $N_{TR_{tm}} \leftarrow N_{TR_{tm}} + 1$
22                 end
23             end
24             $Remove(TR_{tm}, TR)$
25         end
26     end
27 end
28 return $N$
```

between reviews casted by users on different products. At the end of our algorithm 1, we form a network, N, where each vertex, v, represents a user, and the weight of each edge, e, represents the number of times the similarity between two users' reviews are greater than a threshold over one or multiple products.

3.3.2 Design and Overlapping Method

Our implementation of algorithm 1 demonstrates that communities (groups) may be well-described by a model that joins small sub-communities (subgroups). Figure 3.1 shows an individual whose review is detected to be similar to other reviewers. The different colored areas represent different products the reviewer, R_1, has reviewed, and the nodes located inside the colored areas are different reviewers whose reviews are most similar to R_1.

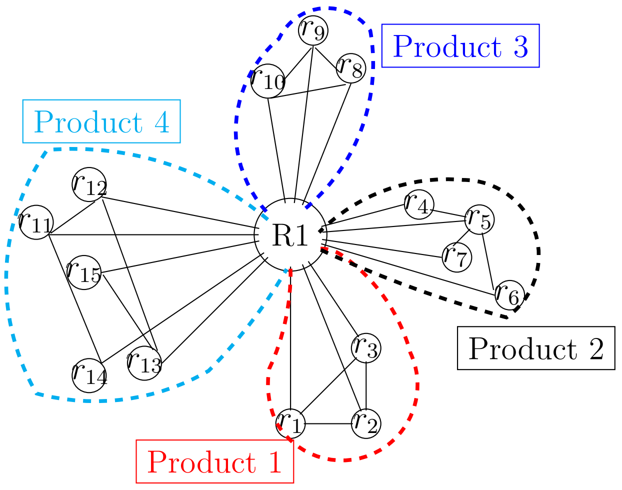


Figure 3.1: A decomposition of the communities.

The detailed description of this method can be explained in two phases. The first phase in this method aims to identify subgroups in a network, N, where edges between users represent the number of times those users have submitted similar reviews on different products. To accomplish this, we iterate over each user, v, in the network, N, and group v's neighbors into subgroups.

A user may belong to many large colluding groups $G_1, G_2, ..., G_k$ each of them containing v and some other reviewers who are similar to v. This can be seen as the first graph in Figure 3.2 where each colored area is a colluding group, G_i, on one product. Each subgroup $SG_{1,v}$, $SG_{2,v}$, ..., $SG_{m,v}$ contain v and vs neighbors that belong to $G_1, G_2, ..., G_k$. Each $SG_{m,v}$ will be derived as seen in the second graph of Figure 3.2. The same subgroup may be created multiple times, therefore $k \leq m$. For instance, when considering subgroups centered nearby user v_1, we may create a subgroup $\{v_1, v_2, v_3\}$. When considering subgroups near v_2, we may also create subgroup $\{v_1, v_3\}$. We prefer to allow multiple copies of the same subgroup to exist, but our method still works even if only one instance exists. The multiple copies allow us to find the overlapping colluders. Relationships between users can efficiently be placed into one subgroup, as the reviews of the followers will always be more like the leader than any other colluder across all products.

In the second phase, we build a new network W such that each node in W represents one subgroup from the first phase. Two nodes in W are joined by an edge if the subgroups that they constitute are interconnected, for instance, sharing users in N. We also require that two subgroups share a set number of users before connecting them in network W. This number is discovered through cross-validation.

Finally, for identifying groups in the new subgroups network, we use an existing community detection algorithm to group the elements of W. To determine the colluding groups in N, we look at each group G that was produced by W. We take the union of all users from N that are included in the subgroups represented by the users in G. Note that because each user v may appear in multiple subgroups in the second phase, v may appear in various groups in W. Requiring that a user v from

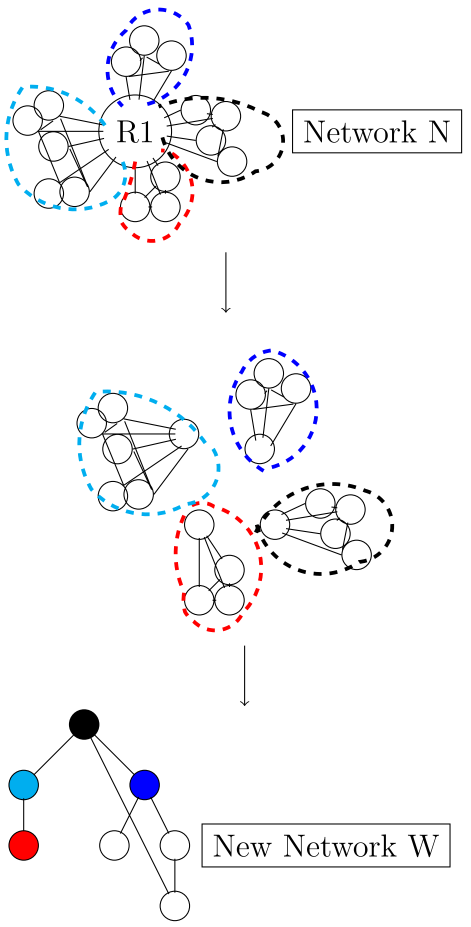


Figure 3.2: A reviewer who is copying from multiple colluding groups.

N appears in at least n subgroups from G (i.e., at least n users from N have similar reviews with v in community G), results in smaller, more tightly-knit groups, with the possibility that some users from N will not appear in any groups.

The method we use here is different from others for two main reasons. First, other methods use typical algorithms that have a rigorous concept of subgroups. For instance, the algorithm Clique Percolation used in [KCJ14] requires sub-groups to be cliques of fixed sized. In our method, we use a community detection method to discover sub-groups. The main advantage in this community detection is that it is not dependent on fixed clique sizes and may identify more subgroups than Clique

Percolation. Second, other methods generally connect sub-groups by identifying connected components between those sub-groups. Our definition of connectedness is the defining part that will lead to better identification of sub-communities. We connect components in our graph only if they are similar across different products.

3.3.3 Overlapping Collusion Detection Method

The purpose of the previous section was to transform a list of reviews into a network of collusion. Now we present our second algorithm to identify the overlapping groups. We use the ideas from [SH15], that uses existing community detection algorithms to identify more realistic communities by first discovering local sub-communities and then determining all the communities in the network. For us, we find subgroups and observe that each user can appear in many subgroups allowing us to detect overlapping colluders. In this algorithm, we performed the first and the second phase of the method to identify subgroups in the network, N, by using the subgraph, K_v, formed by v's neighbors. Then we apply the Louvain greedy modularity on each K_v subgraph to partition v's neighbors into disjoint sets. To obtain subgroups, we add v on each of these sets.

We build a new network, W, where each node represents one subgroup from the first phase of algorithm 1. Two nodes in W are linked together if they have a Jaccard similarity larger than a predefined threshold between them, i.e., two nodes had several similar reviews across multiple products. This threshold is a hyperparameter, which means that we select it based on which value of the hyperparameter gives the best accuracy. The weight of an edge in W will be the Jaccard similarity between two adjacent nodes. Similarly, on W we apply Louvain greedy modularity optimization to divide the nodes in W. To determine the overlapping colluders in N, we search

Algorithm 2: Overlapping collusive groups detection algorithm

```
Input  : Network $N$ with vertices $V(N)$ and edges $E(N)$.
Output: Overlapping groups in $N$.
       : $SG$ multiset of sub-groups, empty initially.
       : $AOG$ Set of all overlapping groups detected, empty initially.
1  for $\forall\ v \in V(N)$ do
2      $K_v \leftarrow \{y : (y, v) \in E(N)\}$
3      $T \leftarrow Louvain(K_v)$
4      for $\forall$ set $S \in T$ do
5          $S \leftarrow S \cup \{v\}$
6          $SG_i \leftarrow S$
7      end
8      Initialize network $W$, where $\|V(W)\| = \|SG\|$
9      Defined $f : V(W) \rightarrow SG$
10     for $\forall\ s \in V(W)$ do
11         for $\forall\ t \in V(W)$ do
12             if $SimilarityJaccard(f(s), f(t)) \geq \theta$ then
13                 $E(W) \leftarrow (s, t)$
14             end
15         end
16     end
17     $Z \Leftarrow Louvain(W)$
18     for $\forall$ set $Z_i \in Z$ do
19         Initialize list $L$
20         for $\forall\ s \in Z_i$ do
21             $L \leftarrow L \cup f(s)$
22         end
23         $AOG_i \leftarrow L$
24     end
25     return $AOG$
26 end
```

through all the groups created in the new network W. For each group, G, in W we take the union of all users from N that are included in the subgroups represented by the users in G. A set of overlapping colluders will be produced on each iteration. This is the same network W presented in the last phase of Figure 3.2.

3.4 Simulations Results and Analysis

In this section, we introduce the dataset used for this work and present the simulation results to validate the effectiveness of our defense mechanism.

3.4.1 Simulation Setting

The dataset used to test our algorithm consists of 1600 reviews from the 20 most popular hotels in Chicago [OCCH11], [OCH13]. The reviews are split into groups of truthful and deceptive comments made on each hotel. Half of this dataset (deceptive) was created through AMT crowdsourcing, where they ask each of the reviewers to write deceptive (positive/negative) reviews on each hotel. In this dataset, reviewers were restricted to one submission, and each review must be at least 178 words.

The purpose of this work is to detect overlapping groups of colluders with different sizes across multiple product reviews. Therefore, we must take some further steps to get the features of the review text to discover those colluders on this dataset. For the community detection method that we used, in addition to this standard dataset, we added some non-adversarial reviews on different hotel reviews casted by different users. We then removed the stopwords from all the reviews and used a POS tagger [TKMS03] on each review to extract only POSs for processing.

The non-adversarial reviewers represent those users who do not want to work to earn money and prefer to work between them on different hotel reviews, and their goal is to gather remuneration without doing the actual task.

3.4.2 Interpretation of Results

The purpose of this section is to present how well the proposed methods performed in detecting overlapping groups of colluders on reviews casted on multiple products through crowdsourcing. The Algorithms were evaluated in terms of Precision, Recall, and Accuracy. Precision can be defined in terms of true positives (TP) and false positives (FP) as mentioned in equation 3.5 [DG06]. Recall is the result of true positive compared to false negative (equation 3.6) [DG06], while Accuracy is the ratio of true results compared to the total number of cases examined (equation 3.7) [DG06].

$$precision = \frac{TP}{TP + FP} \tag{3.5}$$

$$recall = \frac{TP}{TP + FN} \tag{3.6}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

The definition for each condition is as follows:

True positive (TP): result indicates that given reviewers are indeed detected as colluder and also associated the colluder with the correct sub-community.

True negative (TN): result indicates that given reviewers are indeed detected as legitimate.

False positive (FP): result indicates one of two things. First, that given reviewers are mistakenly detected as colluder and place them in the wrong sub-community. Second, that given reviewers are mistakenly detected as colluder.

False negative (FN): result indicates that given reviewers are wrongly detected as legitimate.

Note that in our definitions we choose a strict condition for false positive. This is because our main goal is to find the colluders and to classify them in their correct sub-communities. We intend to find as many real colluders as we can and place them in their proper groups(true positives). We also want to minimize the number of real colluders identified and not placed in their correct groups (false positive). That is to say, we want to maximize our precision in order to find the groups of people writing fake reviews. However, because a monetary reward is involved, we are punished when identifying a normal reviewer as a colluder. A recall which is low will discourage reviewers from using such crowdsourcing services.

Precision and recall are closely related to each other, so an algorithm with a high recall normally has a low precision and vice versa. Our method investigates first on potential groups of colluders on reviews casted on different products. Second form a network N to identify subgroups, where the edges between two users represent the number of time those users have submitted reviews with similarity greater than the hyperparameter threshold on different products. Third, we built a new network W such that each node in W represents one subgroup, two nodes in W are joined by an edge if the subgroups they constitute are interconnected, for instance, sharing users in N. By using our dataset [OCCH11], [OCH13], we modify the value of the threshold parameter into small variations to attain a deeper knowledge of how delicate it impacts a trade-off between precision and recall. To gain insight into the sensitivity of the threshold parameter of our FuzzySim to any small change of its value, we also plot its influence on precision, recall, and accuracy. The figures below show how our similarity performs on the detection of similar reviews on Ott dataset [OCCH11], [OCH13] with non-adversarial colluders.

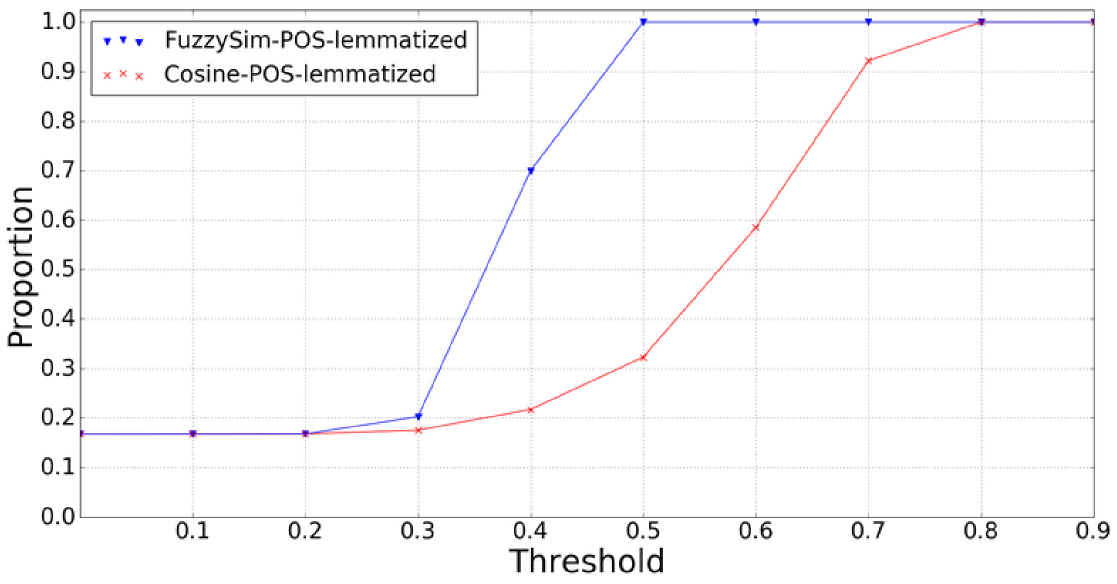


Figure 3.3: Detection of Similarity between Reviews (Precision)

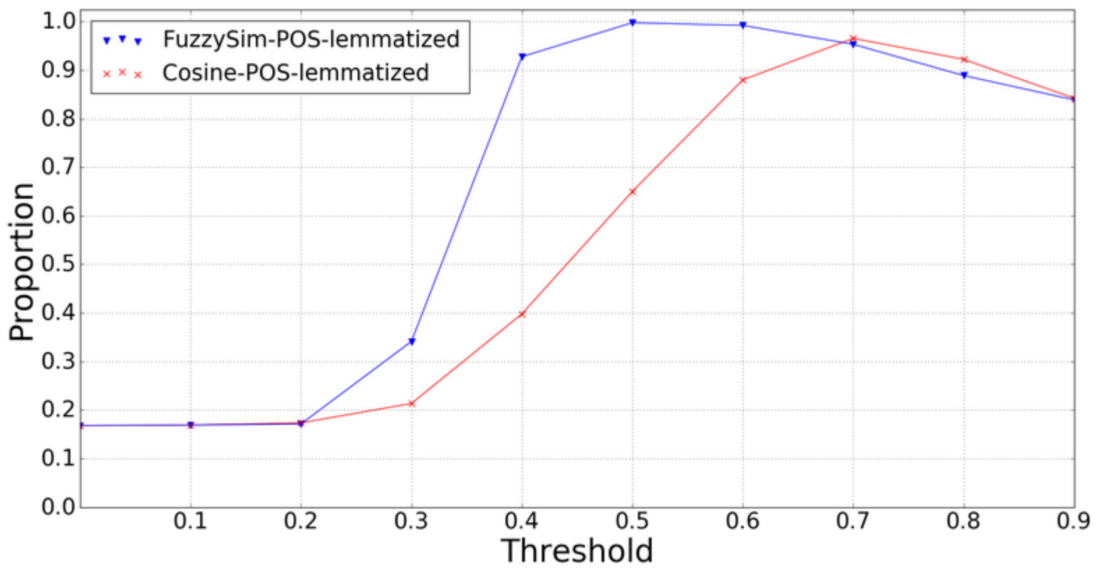


Figure 3.4: Detection of Similarity between Reviews (Accuracy)

Figure 3.3 and figure 3.4 show the performance of Algorithm 1 for the detection of similar reviews using the FuzzySim (blue) against the vectorial cosine similarity (red) proposed in [SE15], both similarities include POS tags and lemmatization. Our FuzzySim with POS tags and lemmatization achieves a higher precision and accuracy as the threshold is raised. This is proven by the results, Figure 3.3 shows that our FuzzySim achieves the highest precision (100%) at a threshold of 0.5 while the vectorial cosine achieves it at a threshold of 0.8. The highest Accuracy (100%) is achieved at a threshold of 0.5 by our FuzzySim, while the vectorial is at most at 97% at a threshold 0.7. Overall our FuzzySim outperforms the vectorial similarity.

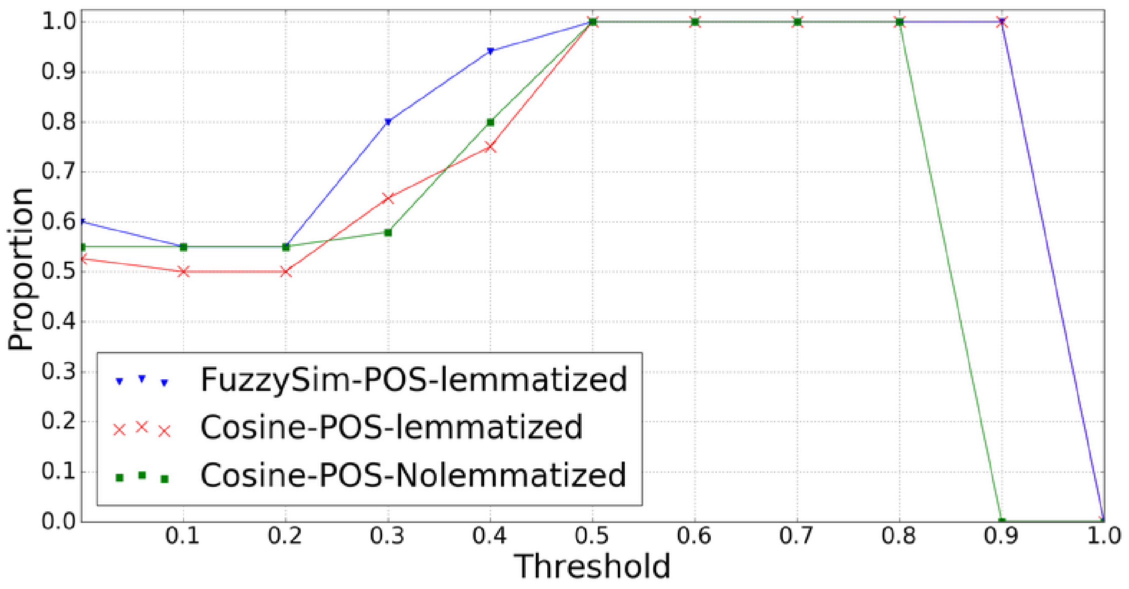


Figure 3.5: Overlapping Colluding Detection Performance (Precision)

One important thing to note is that as the threshold increases, the recall does in fact decrease. Therefore the method of comparison which can achieve the higher precision while minimizing the loss of recall more follows the goal mentioned before for the same reasons. Also, false positivity for algorithm 1 has only one case, the algorithm classified a reviewer as a non-colluder and is in fact a colluder.

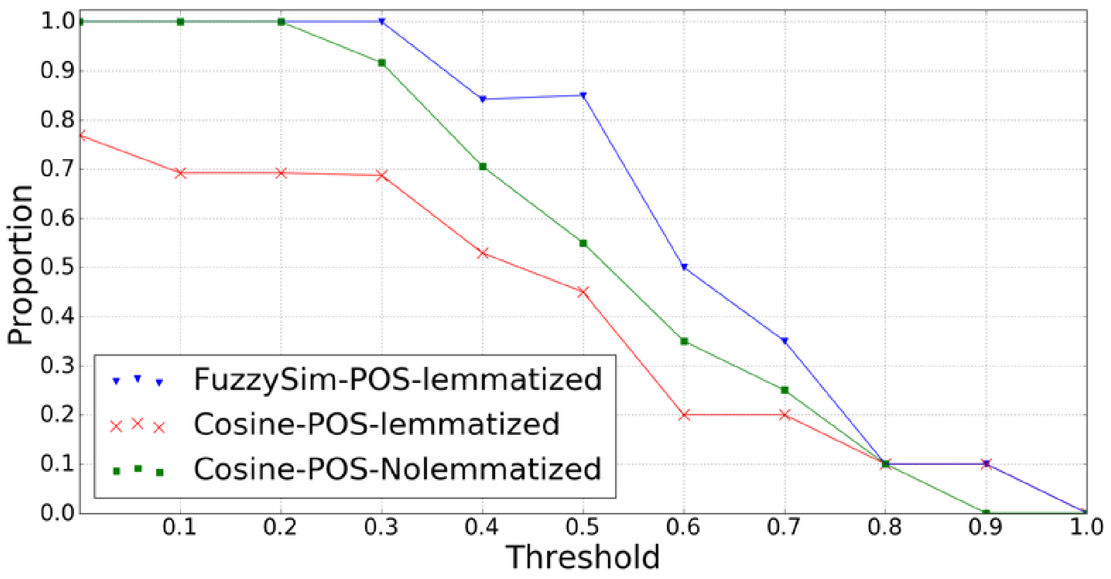


Figure 3.6: Overlapping Colluding Detection Performance (Recall)

The evaluation of Algorithm 2 for the detection of non-adversarial overlapping groups of colluders with different sizes of cliques is presented in Figure 3.5, figure 3.6, and figure 3.7. We again compare our method of comparison with those used in [SE15]. We also use an additional comparison by leaving [SE15] method without lemmatization to show how much more effective it is.

The FuzzySim used in algorithm 2 outperformed the vectorial similarities once again. Its precision increases faster starting at the threshold of 0.2 to reach 100% (no false positives) at a threshold of 0.5. The recall of our FuzzySim is overall higher than the other methods.

In every performance test, the FuzzySim outperformed the methods mentioned in [SE15]. In these results, we focus more on our Accuracy (Figure 3.7) over individual Precision (3.5) or Recall (3.6) to determine the best threshold to detect overlapping communities. This is because in a network where many non-adversarial colluders exist, the stricter the threshold the more likely we find these colluders. However, we do not find many. According to our accuracy, the threshold of 0.5 would be

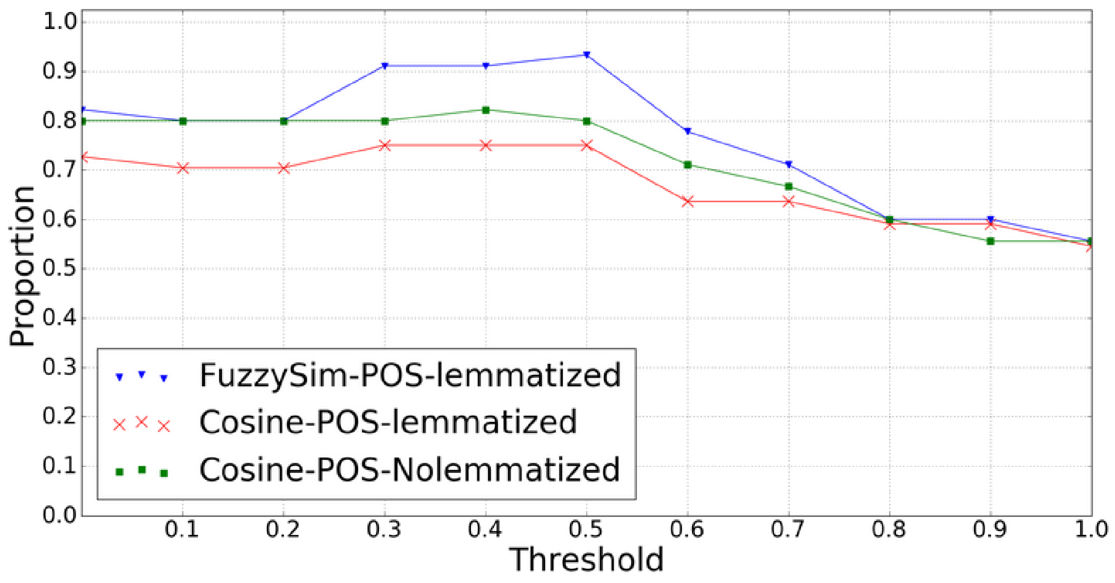


Figure 3.7: Overlapping Colluding Detection Performance (Accuracy)

best to use because it achieves the highest recall and precision. Out of the vectorial similarity, the cosine with POS and without lemmatization achieves a better result compared to the cosine with POS and with/without lemmatization for the highest accuracy at a threshold of 0.5.

We believe the Cosine method does not benefit from the process of lemmatization because it is rather strict on the comparison of individual tokens. Also, as mentioned previously the comparison of prepositions with nouns/adjectives is rather important. A vectorial comparison will place more importance on commonly used prepositions whereas our method purposely chooses to ignore them.

3.5 Summary

In this chapter, we introduced the problem of non-adversarial overlapping colluding in crowdsourcing. This problem is challenging because these overlapping groups are not necessarily the same size, and an approach which has low recall will discourage normal reviewers. In contrast, we propose a hybrid similarity with POS and lemmatization for detecting collusion review in crowdsourcing that outperforms vectorial similarities [SE15] which were proposed to solve this issue. We proposed two methods to detect overlapping colluding in crowdsourcing [KPI$^+$17b]. First, we proposed an algorithm using the FuzzySim that performs better than the vectorial similarities [ME$^+$96, SE15, WLF11, Jar95, Jar89]. Second, we proposed a method that employs an algorithm built in community detection to find overlapping colluders with different group sizes over different product reviews. Finally, we used Precision, Recall, and Accuracy to show the quality of the output of methods. Limitations and future directions will be discussed in Chapter 7.

CHAPTER 4

COLLUDING IDENTITY CLONE ATTACKS IN ONLINE SOCIAL NETWORKS

Online Social Networks (OSNs) have become one of the most common ways for people to communicate: consequently, this has given attackers a new way to steal information from influential users. This has brought new forms of attacks to take advantage of users trustworthiness in OSNs. This exploitation leads to attackers who combine both classic and modern threats. Specifically, colluding attackers have been taking advantage of many OSN users by creating fake profiles of their friends and attempting to exploit them in other OSNs where they may not have accounts. Colluders impersonate their victims and ask for friend requests to the target in an attempt to infiltrate their private circle to steal information. This type of attack is difficult to detect in OSNs because multiple malicious users may have a similar purpose to gain information from their targeted user. In this chapter, we give an introduction in Section 4.1. After, our problem statement will be presented in Section 4.2. In Section 4.3, we described our system model. After that, the treat model is explained in Section 4.4. The detailed proposed solution and the interpretation of the results are introduced in Section 4.5 and Section 4.6, respectively. Finally, a summary will be given in Section 4.7.

4.1 Introduction

In recent years, Online Social Network (OSN) services have rapidly grown into an integral part of society and offer users a variety of benefits. Users see these OSNs not only as a platform to establish contacts but also as a means of soliciting contributions from large groups of people, exploiting various applications existing in OSNs. However, OSNs also bring new threats to the community. OSNs became a leading weapon for launching cyberattacks on any organization and its people [Cis18]. In 2015, the Twitter accounts of U.S. military officials were hacked by those claiming to support the Islamic State [Ros15]. In 2017, Iranian hackers took control of social media accounts of U.S. State Department officials [SP15]. Furthermore, a new directive policy was signed in May 2016 by the Director of National Intelligence to allow investigators to gather publicly available information on social media from people who are undergoing background checks for security clearance [Aft18].

One original attack that is essentially different from the environment and infrastructure of OSNs to compromise user is called Identity Clone Attack (ICA). The main purpose of this attack is to clone an existing user in order to infiltrate a particular social circle to gather information normally shared to the trusted users [BSBK09]. These attacks can be used across multiple OSNs to further confuse the targeted users, and this process is called cross-site profile cloning. This work aims to build a defense mechanism that can detect whether a user's friend request is a clone and find the colluder's accounts. Admiral James Stavridis, NATO's most senior commander, was affected by identity clone attack. It was reported [Aft18], [Lew12] that hackers set up fake Facebook accounts under his name in the hope that his colleagues, friends, and family would make contact and answer private messages, to gain sensitive information about him or his network. Our work also takes into consideration different attack patterns, including colluding attackers who recreate social circles of the targeted user to further confuse the target and send multiple friend requests to the same user. The majority of people in OSNs tend to use the default privacy settings, and several of those users have similar names in the real world. As a result, the profiles in OSNs tend to be similar as well. Therefore, we need to consider more attributes than just names, to avoid the assumption that profiles with similar names are considered fake.

In this chapter, we propose a three-step method to detect ICA colluders in OSNs. Our method matches two user profiles from two different OSNs based on classification techniques that use features extraction on a users' friend request information and on users' friend lists. We used a binary classification to rank the probability that multiple users' friend request from different users can be colluders. This classification primarily utilizes two types of features: (1) user's general information and (2) friend network similarity, such as the number of mutual friends between the requesting user and the target. This method consists of three steps. The first step is to gather profile information of the users' friend request and search through the same or other OSNs for profiles that are similar and returns the accounts that are the most similar based on the information gathered. The second step verifies the identity of each of the user friend request through the friend list. The third and final step returns which of those friend requests are colluders. Unlike previous works, our method uses a hybrid-based, string matching similarity algorithm to find profile similarity, and it achieves high accuracy detection on different OSNs [CRF03]. This is because it is a recursive matching scheme used for comparison. It also includes an internal similarity function for token-based comparisons.

4.2 Problem Statement

The behavior that people present while using online social networks has brought those platforms in the light of malicious users, who are now seeing them as a significant way to initiate/pose threats to different users that are within/outside an organization. Therefore, a strong defense mechanism against collaborative malicious users needs to be addressed. In the following, we aimed to answer the following question:

Can the profile attributes improve the detection mechanism of ICA collusion in OSNs?

We hypothesized that token-based similarity and friend list structure could uncover malicious in ICA collusion with high accuracy. In colluding identity clone attack, we aim to uncover all the malicious clones that participate in a collaborative attack, called collusion. We used a hybrid-based similarity algorithm for its high efficiency in detecting similar profiles.

Despite using profile attributes based on a hybrid-based similarity algorithm and friend list similarity to weaken ICA collusion, designing a defense mechanism poses a new difficulty. Colluding ICA may want to be undetected by defense mechanisms in place; therefore, colluders can add or delete some profile attributes on their clones to make them look legitimate.

4.3 System Model

We have a set of n OSNs with $S_n = (X_n, Y_n)$, where X_n is the set of nodes and Y_n is the set of edges connecting the nodes together. Each node $x_i^n \in X_n$ is an individual x_i in a social network, S_n. An edge $y_{(i,j)}^n \in Y_n$ presents the relationship between x_i^n and x_j^n. The attributes that will be considered here are the following: Name (x_{name}^n), education ($x_{education}^n$), city (x_{city}^n), age (x_{age}^n), workplace (x_{work}^n), gender (x_{gender}^n), and friend list (x_{fl}^n). When each user receives a friend request(s) from another user(s), the framework will take as input the attributes/features provided by the requester(s) to detect the similarity between each pair of attributes in OSN$_2$ and OSN$_1$. The framework then produces the most similar profile between the friend requester profile(s) and the profile that is a potential match. In an ICA attack, the attacker will need to clone an already existing user on the same or different OSN.

To make our method more accurate, we assume that the friends in the user friend's list are legitimate. The aim here is to detect and prevent ICA colluders from having access to private information of their target.

4.4 Threat Model

We know that in the ICA attack, the colluding attackers know the identity of their target [BSBK09], [JLTJ12]. This includes knowledge about the real social network of the target (e.g., the colluders may know the people with whom the target usually cooperates in real life, friends, coworkers, family members). The colluders will use the attributes that specific victims put on their profiles to impersonate them in the aim to steal the trust that the target and his private circles have amongst themselves. The objective of the colluders is to clone victims (profiles) that are direct friends of the target's direct friends (e.g., friends that are one hop away of the direct friends of the target) to avoid detection by the target [HCSS14].

All colluders clone their victim profiles and avoid detection by behaving in two ways. First, the colluders must identify and be sure not to create cloned profiles of direct friends of the target because this could attract the attention of the target that could deny the requests. If the colluders try to clone the target's direct friends in the same or different social network, the target can use a challenge (or instant chat) to verify if the friend request is genuinely coming from their private circle [HCSS14]. Second, in order to build trust among the target's circle, the colluders must choose specific victims (from OSN_1) to clone in OSN_2. For instance, even if those victims exist in OSN_2, as long as they are not direct friends of the target, the colluders can clone the profile and submit a friend request to the target.

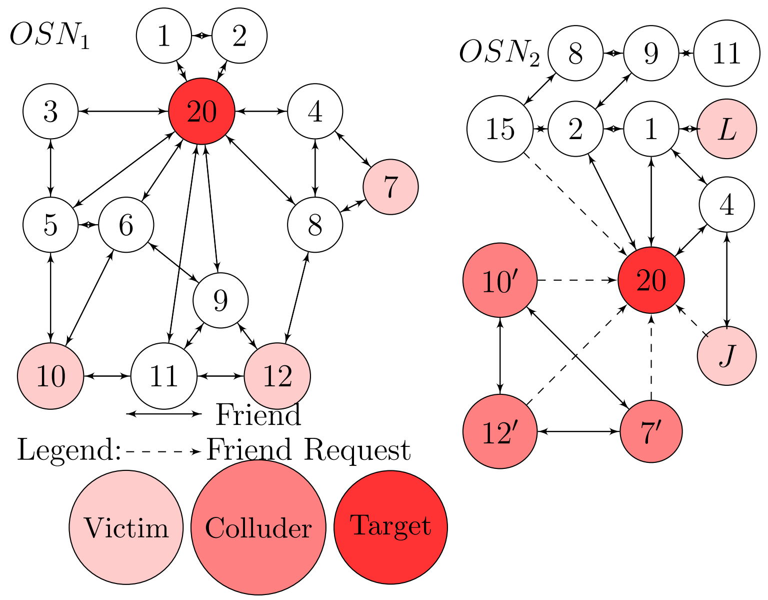


Figure 4.1: Colluding ICA.

To have a higher percentage of success, the colluders will cooperate among themselves by accepting each others' friend request and submit a friend request to the same target. In addition, we consider that the colluders are intelligent, aware of the list of victims' attributes, and know the friend list of the victims.

Figure 4.1 displays an example of how an ICA attack with colluders would work. The private information flow between friends in OSN_1 is the target of a group of attackers who will cooperate with each in order to have a higher chance of succeeding. These colluding attacks can be difficult to detect and harmful for the legitimate user. In Figure 4.1, attackers gather public information from the victims $(10, 12$ and $7)$ in OSN_1 and create fake profiles of the victims in OSN_2 (i.e., $10', 12'$ and $7'$). They make fake profiles look legitimate by accepting friend requests among themselves and sending friend requests to the same target belonging in the new social circle formed in OSN_2. For this type of attack to succeed, the attackers will not create fake profiles of direct friends of the target $(1, 2, 3, 4, 5, 6, 8, 9,$ and $11)$; otherwise, the attack has a lower chance of success [HCSS14].

A real-life application of this is as follows. The social friends $(1, 2, 3, 4, 5, 6, 8, 9,$ and $11)$ formed in OSN_1 can be directors of some services in a well-known technology company Alpha that shared information about the company. The people who clone the victims $(10, 12,$ and $7)$ in OSN_2 can be associated with other technology companies Beta, Gamma, and Sigma.

A collaborative attack between malicious profiles will allow each member in the group to have a higher probability of succeeding when the target accepts their friend request. The colluders will then share the information they gain among each other. In this scenario, the attackers will send their friend requests together to the target. This collaboration between attackers will maximize the probability of success that each member would not be able to achieve individually.

Therefore, for the attackers $(10', 12',$ and $7')$ to have information about Alpha's new products, they are likely to have a collaboration between Beta, Gamma, and Sigma in an attempt to gain the private information. The attackers in OSN_2 will each add to their friend list the victims' direct friends related to the target. For instance, $10'$ will add $5, 6, 11, 12', 7'$ as friends in his or her friend list in OSN_2, and so on.

4.5 Proposed Method

The main idea behind the proposed approach is based on the social network interactions of users with other friends, where colluders will try to avoid detection by the legitimate user(s). Our approach will uncover many cloned profiles that are submitting friend requests to an influent user. In this model, we assume that the clones are mutual friends. We propose a method composed of three main components, and each part is detailed in following subsections. Remember that the malicious profile detection in social networks is a task represented by a binary classification task. So, this process would begin with data initialization, followed by the training set. The training set is used as an input to the method that induces a classifier. The classifier estimates the probability that several users' friend requests to a target are colluders. The goal here is to discover the biggest possible number of members of an ICA colluding in social networks.

Data Initialization

To detect ICA colluders across social networks, a large and appropriate dataset from OSNs is required. Nevertheless, it is hard to find datasets with textual information based for ICA attacks to try and find colluders to validate our framework because all are anonymized [VMCG09a]. We do not validate our framework detection in a real system because our detection framework needs textual attributes to be validated. Attributes such as profile pictures were not in the scope of this paper because it requires a different similarity metric. We created a synthetic dataset of 2000 profiles with attributes. We also created social circles and friend lists between each user (random graph) in our data in order to make it similar to Facebook and Google+, where each OSN has 1000 profiles in their network. We do not model content-oriented sites such as Twitter, YouTube, or Flickr because these contain some features of OSNs but are rather microblogging sites or content communities with different characteristics than OSNs [PGP13].

At the beginning of each simulation, we randomly generate the social network where we create multiple targets of influence. These targets will tend to have a much larger friend list than regular nodes in the network. For each of the friend connections that are generated for targets of influence, we give a preferential probability for them to form cliques among themselves. For all the other nodes, their friends and friend connections are randomly generated. After generation, a target and then a number of colluders are picked randomly from a preset range. These colluders are selected using the behavior described in the previous section and as shown in Figure 4.1.

Dataset

In this phase, the goal is to identify which users belong to both social networks. We performed an identification of each user's profile attributes. There are two main types of features: (1) user's general information and (2) friend network similarity. We used 1500 matching profiles from both OSNs as our training set, and then we tested the remaining 500 profiles for validation. Note that this is not necessary for profiles to exist in both OSNs.

We generated victims based on the following concept: attackers are intelligent, and the target will be among one of the user profiles who has the most friends. In an effort to avoid detection by the framework, the attackers will slightly modify some of the public attributes (name, education, city, age, workplace, gender, friends list) of the victims and try to recreate a social circle with the target.

Structure of the Model

The classifier takes two sets of inputs. First, it takes in the results of the search for the most similar profile making the friend request, with a percentage for each category as defined before. The second input is the profile whose friend list structure most closely matches the current profile making a friend request to the target. We wish to separate the actions of regular profiles and the percentages they will have from profiles that are cloned. As mentioned before, if features are missing in the comparison, the classifier we chose will not be bothered by it.

The detection will rely on the fact that two or more cloned profiles will try to impersonate and infiltrate a social circle. We make the assumption that colluders must have some of the public attributes belonging to the victims. The process of detection happens in three stages. First, with these attributes our algorithm searches through the OSN where the friend requests have been sent. The algorithm returns similar profiles that have the closest similar attributes between both OSNs. Second, it will compare the friend list similarity to return the closest ones. Third, to make our classifier robust, we consider that there are genuine profiles among those asking friend requests to a target. That is to say, if we find one colluder among those making a friend request, we cannot assume the rest of the requesters are also colluders. In our simulation, we made a random amount of genuine profiles that also ask for a friend request to see how well the algorithm performed and to see if it was overfitting.

Attribute Similarities

Monge and Elkan [ME$^+$96], [ME97] proposed an effective and simple recursive matching scheme to compare the similarity between two long strings, A and B. First, A and B are broken into several tokens $A = a_1, a_2, \ldots, a_i$ and $B = b_1, b_2, \ldots, b_j$ with $i = \|A\|$ and $j = \|B\|$, respectively. The similarity between A and B is defined as

$$Sim_{MongeElkan}(A, B) = \frac{1}{\|A\|} \sum_{i=1}^{\|A\|} max\{sim'(a_i, b_j)\}_{j=1}^{\|B\|} \tag{4.1}$$

where sim' is an internal similarity function able to measure the similarity between two individual tokens.

For the internal similarity function, we use Jaro-Winkler distance [Jar89], [Jar95]. This is based on the number of common characters m between two strings and the number of transpositions t as follows:

$$Sim_{Jaro\_Winkler}(a,b) \begin{cases} sim_{jaro}(a,b) & if\ sim_{jaro}(a,b) < b_t \\ sim_{jaro}(a,b) + \frac{l}{10}(1 - sim_{jaro}(a,b)) & otherwise \end{cases} \tag{4.2}$$

with

$$Sim_{Jaro\_Winkler}(a,b) \begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}(\frac{m}{\|a\|} + \frac{m}{\|b\|} + \frac{m-t}{m}) & otherwise \end{cases} \tag{4.3}$$

where l is the length of common prefix at the start of the string up to a maximum of 4 characters, and b_t is the boost threshold (0.7) in Winkler's implementation [Jar89], [Jar95].

Two characters, a and b, are considered matching or common only if they are in the sliding window size and not farther than $\lfloor \frac{max(|a|,|b|)}{2} \rfloor - 1$. The number of matching (but different sequence order) characters inside of the window size defines the number of transpositions. For instance, in comparing CREATE and TRACEE, 'R', 'A', 'E', and 'E' are matching characters, m = 4. Although 'C' and 'T' appear in both strings, they are farther than 2, i.e., $\lfloor \frac{6}{2} \rfloor - 1 = 2$. Therefore, $t = \frac{2}{2} = 1$.

However, the Monge-Elkan method does not follow the symmetric property. Consider example 8 in Table 4.1: the method computes their similarity from the viewpoint of A by taking all the tokens in its string and looking for the most similar token in B. From B's, the viewpoint (example 7 in Table 4.1) we will have a different result. So, this asymmetric property will lead us to inconsistent results for practical problems when the two strings have a common or different number of tokens because an attacker can use this weakness to clone a profile successfully without detection.

Because existing similarity functions have limitations, to address these problems, we propose a modified similarity metric called FuzzySim that will overcome these limitations and outperform other metrics. FuzzySim is defined as follow:

$$FuzzySim_{MongeElkan}(A,B) = \frac{1}{max(\|A\|, \|B\|)} \sum_{i=1}^{\|A\|} \bigcap_{Average\ \theta} \{sim'(a_i, b_j)\}\|_{j=1}^{\|B\|} \quad (4.4)$$

We used a hyperparameter, $\bigcap_{Avrg\theta}$, instead of the max method in equation 1 because it will always return true for strings that are partially matching. An example of this is shown in Table 4.1. In addition, we implement a max string length $max(\|A\|, \|B\|)$ to overcome the Monge-Elkan asymmetric property for the practical problem of comparing names (see Table 4.1 below, example 7 and 8, to see symmetric).

Our similarity method (equation 4.4) uses a hybrid similarity, of both token similarity and character-based similarity, which considers the different ordering, misspelling of strings. For example, a user U_1, creates a profile with the name "Aaron Galdan" and a clone attacker creates a profile under the same name, but to prevent detection, the attacker may manipulate or even reverse the name,"Galdan Aaron" in order to avoid being detected by the algorithm and impersonate the friend user.

Therefore, our hybrid method partitions the string into smaller tokens and removes special characters, commas, periods, and roman numerals. This allows for more precise comparisons for each token part and returns the most similar substrings that are over the hyperparameter defined before. In Table 4.1, we have test cases to compare our proposed string matching algorithm along with other metrics and show that our method outperformed Fuzzy Jaccard [WLF11] and others.

From these test cases, our score outperformed other similarity measures. Table 4.1 also shows the disadvantages of using just character-based or token-based methods.

Names	Fuzzy Sim	Fuzzy Jaccard	Levenshtein	Jaro	Jaro-Winkler	Monge-Elkan
“Martha/ marhta”	0.822	0.000	0.500	0.822	0.822	0.822
“Brandon hernadez/ hernadez brandon”	0.952	0.867	0.250	0.700	0.790	0.952
“Nba macgrady / mcgrady nba”	0.940	0.880	0.250	0.710	0.710	0.940
“Terry maximilian/ Terrance max”	0.876	0.000	0.375	0.700	0.700	0.876
“Lenovo inc / lenovo corp”	0.500	0.333	0.636	0.842	0.905	0.750
“Andrew Wilson / Albert David”	0.277	0.000	0.153	0.534	0.534	0.505
“aaaa / aaaa xaaa yaaa”	0.890	0.333	0.285	0.762	0.857	1
“aaaa xaaa yaaa / aaaa”	0.890	0.333	0.285	0.762	0.857	0.890

Table 4.1: Similarity Test

Friend List Similarity

The main objective of an ICA colluder is to add the victim’s friends that are direct friends of the target and create a trust relationship with them [BSBK09]. However, classifying by profile information alone is not enough, because people can have similar profile information in the real world. Therefore, we must consider the friend list similarity to see how much they are similar, in order to not assume that their profiles are also faked. For that, we used the friend name overlapping count, which is calculated by defining the total number of common friends with matching names in the friends' lists between profiles.

Let $x_c^{m,n}$ be the clone of the legitimate ($x_l^{m,n}$) user either in OSN$_m$ or OSN$_n$, x_{c-fl}^m and x_{l-fl}^m denotes the friend list of the clones, respectively. The legitimate user in OSN$_m$, is similar in OSN$_n$ and can be denoted by x_{c-fl}^n and x_{l-fl}^n. The friend list similarity measure can be expressed as follows:

$$Sim(x_l^{m,n}, x_c^{m,n}) = \frac{\|(x_{c-fl}^m \cup x_{c-fl}^n) \cap (x_{l-fl}^m \cup x_{l-fl}^n)\|}{\|x_{l-fl}^m \cup x_{l-fl}^n\|} \tag{4.5}$$

In equation 4.5, we consider both cases where the legitimate user can be in OSN$_m$ or/and OSN$_n$. We only considered two OSN$_m$ and OSN$_n$ to calculate the friend list similarity. This can be generalized to more than two OSNs if their data are from user-oriented sites such as Facebook, Google+, etc. for our case study. We assume that the attacker will create clone identities that forge friends of the legitimate user and add them in their friend list to appear more legitimate.

4.6 Simulation and Interpretation of Results

The purpose of this section is to present how our classifier can be used for detecting colluders based on the ICA attack explained above.

Classification problems are mostly evaluated in terms of Precision and Recall [DG06]. Here Precision can be defined in terms of true positives (TP) and false positives (FP) as mentioned in equation 4.6 [DG06]. Recall is the result of true positive compared to false negative (equation 4.7) [DG06].

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

Therefore, four different sub - results emerge as follows:

True positive (TP): result indicates that given profiles are labeled to be colluders when the classifier detects as colluder.

True negative (TN): result indicates that given profiles are labeled not to be colluders when the classifier detects the profiles to be genuine.

False positive (FP): result indicates that given profiles are labeled not to be colluders, but the classifier detects colluding.

False negative (FN): result indicates that given profiles are labeled to be colluders, but the classifier detects the profiles to be genuine.

Our purpose is to maximize the number of real colluders found (true positive) while minimizing the number of misclassified real accounts (false positives). A high precision would indicate that we are detecting the colluders well while also not making a mistake of labeling a non-colluder. Recall evaluates how many of the colluders we actually manage to find: if the classifier returns a low recall, then it means it is not efficient in finding colluders. Accuracy measures how well our classifier correctly identifies a set of colluders from a set of genuine profiles.

Precision and recall are closely related to each other, and classifiers with a high recall usually have low precision and vice versa. Our classifier investigates first whether a profile is genuine or has been cloned from an OSN. It then searches those same profiles to see if a similar profile exists with the same friend connection network. Finally, it compares its own network against the network of the target in the OSN. These inputs are used to identify the colluders.

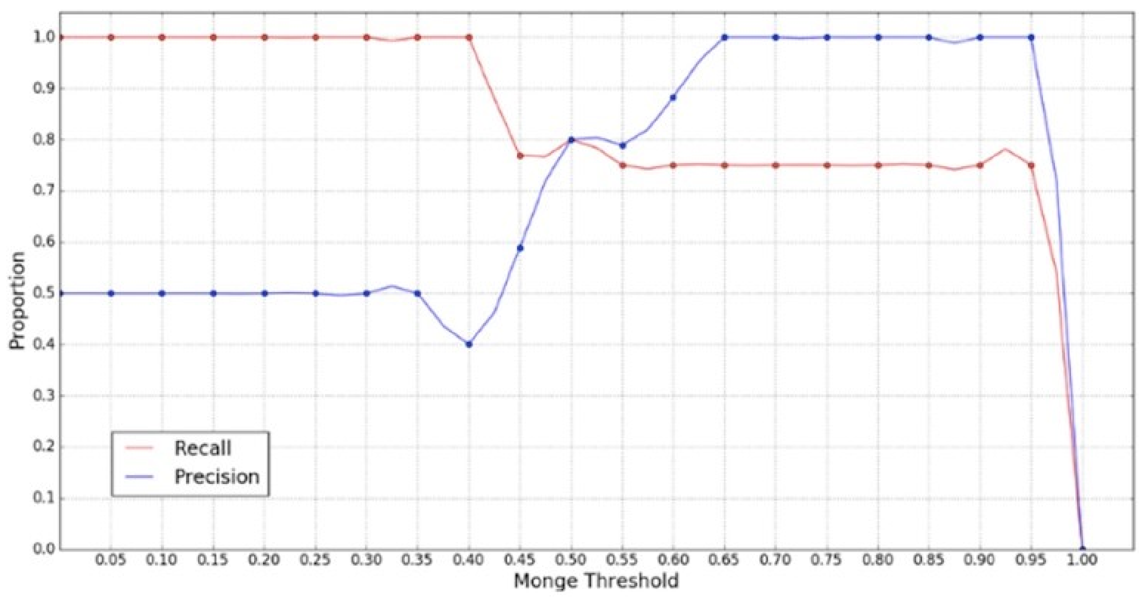


Figure 4.2: Threshold versus Precision and Recall (20 friend request colluders)

By using our labeled dataset, we modify the value of the hyperparameter into small variations to attain deeper knowledge of how delicate it impacts a trade-off between precision and recall. Here, we considered that the number of friend requests is balanced between colluders and truthful requests. To gain insight into the sensitivity of the threshold parameter of our FuzzySim (modified Monge) to the small change of its value, we plot its influence on precision and recall. Figures 4.2, Figure 4.3, Figure 4.4, and Figure 4.5 show the results of our classifier on detecting groups of colluders based on the discussed method with the different number of colluders.

For a better understanding, we know that a classifier with high recall but low precision returns many results; nevertheless, most of the predicted are incorrect when comparing it to the training data. On the other hand, a classifier with high precision but low recall returns very few results, but most of the predicted data are correct when comparing it to the training data. Finally, an excellent classifier will be the one that has high precision and recall, and it will return many results with

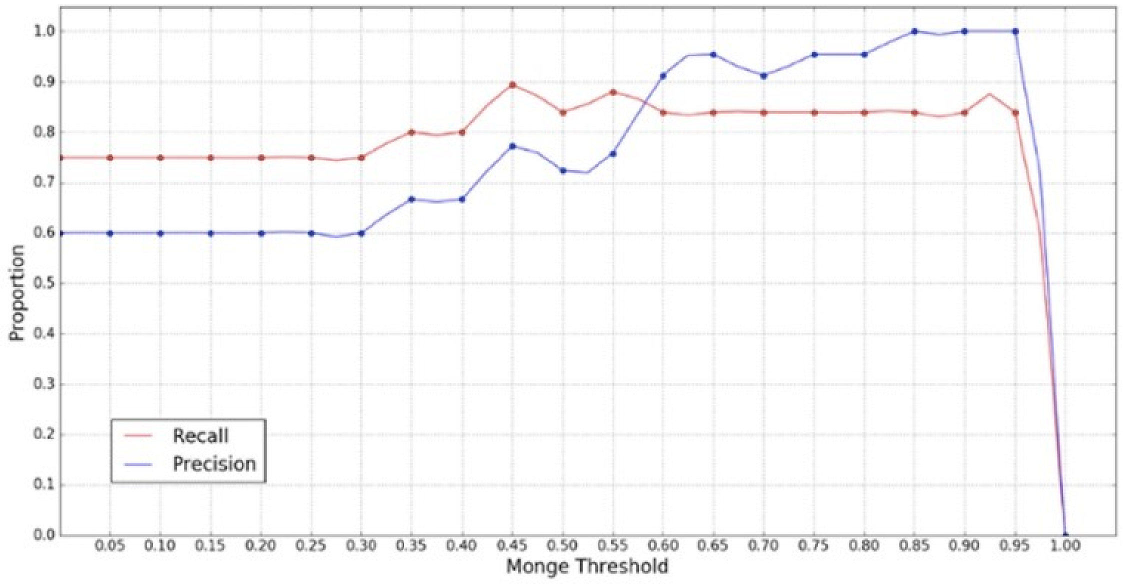


Figure 4.3: Threshold versus Precision and Recall (25 friend request colluders)

all correct results.

According to our goals, the best choice for the hyperparameter will be the highest possible value for recall with precision coming in second. This is because we are interested in detecting the majority of colluders and the penalty for misclassifying genuine profiles is not that large.

Figure 4.2 and Figure 4.3 show our classifier performance with different colluder sizes. Our classifier performs best in terms of precision and recall for 25 colluder friend requests when the hyperparameter is set to 0.5. On Figure 4.3, it has a smoother climb, reaching a recall and precision, respectively, at 90% and 78% for a 0.45 threshold. Both spike at an optimal value of 86% at a similarity threshold of 0.58. In Figure 4.2, both precision and recall reach an optimal value on a 0.50 threshold. Each of those figures is above an optimal value for threshold; for instance, in Figure 4.3, 78% of the colluders are found, and 78% of these are really colluders.

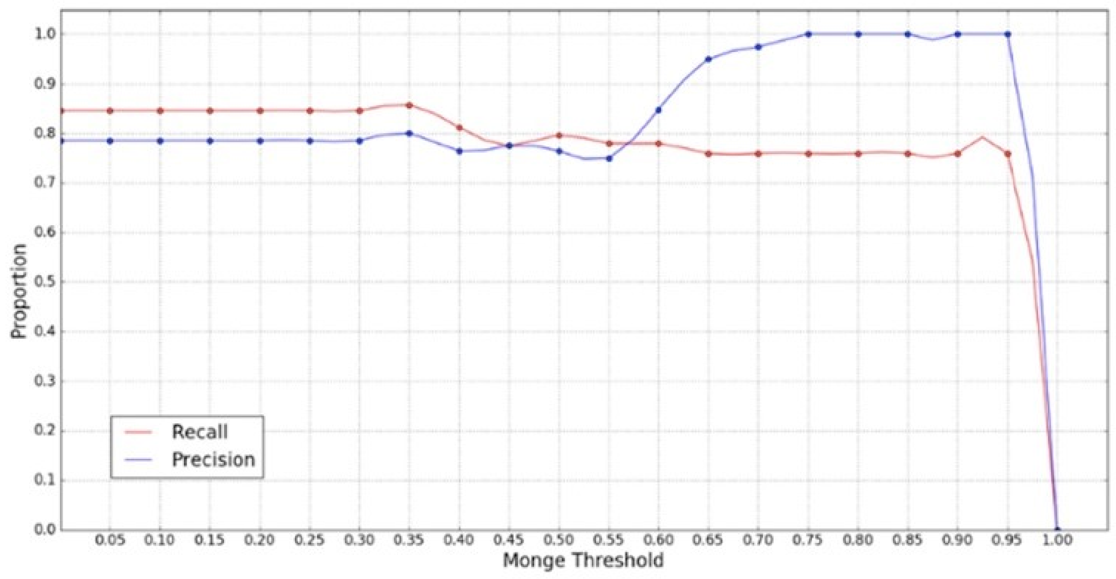


Figure 4.4: Threshold versus Precision and Recall (50 friend request colluders)

For a different insight, we increase the number of friend requests to a target and run the model with varying sizes of groups of colluders to analyze how robust the classifier will be.

In Figure 4.4 and Figure 4.5, we test our classifier performance with bigger colluder sizes. The results show how changing the threshold values impact the quality and accuracy of the classifier. Also, here, we do not specify a specific value for recall and precision, the choice will be made by the user who wants to achieve the highest feasible value for both recall and precision. For instance, in Figure 4.4 and Figure 4.5, the optimal value for the threshold will be 0.565 and 0.575 for Figure 4.4 and Figure 4.5, respectively. In this case Figure 4.4, 78% of the colluders are retrieved and 78% of this result is really colluders. Whereas in case Figure 4.5, 83% of the retrieved colluders just 83% of them are really colluders.

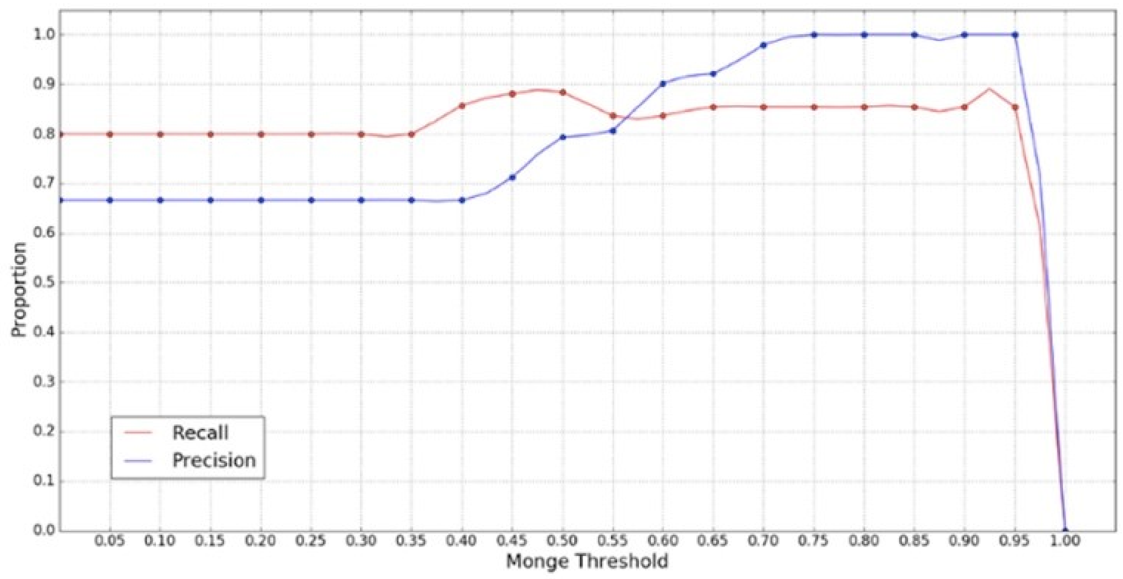


Figure 4.5: Threshold versus Precision and Recall (55 friend request colluders)

4.7 Summary

In this chapter, we introduced the problem of colluding in Identity Clone Attack in Online Social Networks. This problem is challenging because the colluders are able to produce convincing counterparts in different OSNs. Therefore, we propose a learning model to find and match the colluding users [KPI+17a]. Our techniques have the advantage of always being practical because we only use features that are publicly available for each user on OSNs. We then proposed a new type of ICA attack that can be more effective than previous ones [JTJ11, KPIM11, SCL+13]. For detecting such an attack, we introduced three concepts. First, we proposed a modified Monge string metric called FuzzySim that performs better than the original one [ME+96], in order to deal with misclassifying similar profiles and to overcome the problems of exact string matching proposed by previous works [ME+96, WLF11, Jar95, Jar89]. Second, we proposed a friend similarity measure to compare the similarity between friend connection structures. Third, we built a classifier based on FuzzySIm and friend similarity to detect colluders in friend requests. We used precision and recall, to show the quality of the output of our classifier. The limitations and future directions will be discussed in Chapter 7.

CHAPTER 5

COLLUDING TARGETED RECONNAISSANCE ATTACK IN ONLINE SOCIAL NETWORKS

In this chapter, we study the problem of colluding targeting reconnaissance attack that represents possible paths that collaborative malicious users can follow to reach/compromise a specific target in online social networks. We believe that studying and understanding the way attackers proceed, to infiltrate a network without being detected, is the key to develop security preserving countermeasures. Attackers performing targeted reconnaissance attack (TRA) tends to collect precise/personal information of a set of users and organizations for waging attacks by befriending a target user in the future. These attacks consist of adopting diligent means of sending friend requests to users without being detected by current mechanisms. Our goal is to design a colluding TRA's strategy to unveil the target's social circle, utilizing minimum friend requests which pose a new challenge when working with incomplete information on the network topology. In the following, an introduction to this problem will be given in Section 5.1. In Section 5.2, the problem statement about colluding targeted reconnaissance attack will be described briefly. In Section 5.3, the social network model will be explained and then in Section 5.4, the proposed attack scenario will be described and presented. In Section 5.5, we present the advantages of homophily-based similarity metrics on our infiltration process. We present a detailed evaluation of homophily-based similarity metrics used in the attack scenario in Section 5.6. After that, the efficiency and efficacy of our colluding attack strategy will be described in Section 5.7. The analysis and evaluation of our proposed approach are described in Section 5.8. Finally, we give a summary of this chapter in Section 5.9.

5.1 Introduction

Online social networks store vast amounts of valuable data. Therefore, information leaks are disastrous for companies, organizations, and individuals whose data can be skillfully exploited. Moreover, the attackers could be anyone (e.g., cyber espionage, cyber warfare, advertisers for financial gains, and hacktivism) interested in sensitive information [ZG16]. We have seen the growth of various types of attacks in OSNs [FGE14]. One novel yet important attack in OSNs is the *reconnaissance attack* (RA). In an RA, an attacker sends friend requests to users in the network to explore and gain information from the network [ND16]. An extension of the RA is the *targeted reconnaissance attack* (TRA), in which the attacker wants to befriend a specific set of target users to extract private information [ND16]. This chapter analyzes the general collaborative cyber attacks, called colluding TRA, where colluders in TRA tend to share topological knowledge of the network to highly influence their victims. Once a target has been friended by at least one of the colluders, they may then launch more malicious attacks, such as spam [FGE14], spear-phishing [BLM$^+$17], and friend spam [CSYM15] by utilizing the newly obtained information about the target. Consequently, colluding TRA represents a critical link in the progression of OSN attacks.

The underlying basis of TRA is the Triadic Closure theory, which states that users who share many mutual connections present a proper behavior to befriend one another [BMBR11]. Triadic Closure has been shown to apply to OSNs [BMBR11]. The goal of the colluding TRA is thus to befriend friends of the target to increase the odds of befriending the target. To befriend the friends of the target, colluders must first share between them the network topology at each step, then befriend friends of those friends, and so on. By doing so, colluders naturally and suspiciously build their friend network, avoiding detection from defense mechanisms. Specifically, colluders will not build isolated friend groups and risk detection by Sybil defense mechanisms [BBR13].

However, if one of the colluders sends too many friend requests to reach the target, defenses mechanisms used by social network providers may become suspicious or alert. Thus, there exists a trade-off between slowly progressing through the network and sending too many friend requests to reach the target. Furthermore, the colluders will take advantage of their shared knowledge on the network topology to overcome the network visibility restrictions set by OSN providers. For example, the default privacy setting for many OSNs such as Facebook is "friends-of-friends". In a graph-based approach, users can only see other users that are two edges away. This restriction of topological visibility means that colluders must not only befriend the target but also find a path to the target. Compounded with limited friend requests, this presents increased difficulty in conducting colluding in TRAs.

Current research on TRA utilize several different approaches including iterative crawlers [ND16], adaptive information benefit maximization [LSDT16], and batch friend requesting [LST17]. These attacks all rely on network topology for their proposed information benefit metric to help direct the attacker towards the target. However, none of the current work has provided a function that considers the sociological theory behind friendship formation or the colluding behavior that can be used by malicious users to easily find a path to the target by evading detection mechanisms. We advance the research on TRA by developing an information benefit metric utilizing homophily while performing a colluding behavior in OSN. Homophily is the tendency of users to preferentially form friendships with other users sharing a certain degree of similarity in interests and features [MSLC01]. It has been found that attributes such as common hobbies/interests, the closeness of mutual friends, schools attended, etc. significantly influence users' decisions regarding friend requests [RBJB14]. By utilizing user profile information, we leverage homophily to compute information benefit.

Additionally, we describe a new attack strategy based on our information benefit metric. We modularize colluding in TRA into two components. Due to topological visibility restrictions, colluders must first gain visibility before they can befriend the target. We focus on the first visibility portion, which is more difficult because the target's location is unknown. Once the target has been found, it is easier to build mutual friends with the target and a different strategy can be employed. Since each accepted friend request updates the network topology, visibility, and dynamic, we utilize an iterative, adaptive approach. This approach updates with each friend request acceptance and balances between exploration and exploitation to ultimately gain visibility of the target. Furthermore, we consider a maximum friend request threshold that colluders in TRA can send.

5.2 Problem Statement

Online social networks flag users who suspiciously send a large number of friend requests and whose online behavior is anomalous to other users. Though colluders have the ability to send friend requests to any user in the OSN, an exhaustive attempt to directly befriend the target or locate and befriend the target's friends is impractical. Then, we want to provide an answer to the following question:

Can information metrics based on homophily and the shared network topology between colluders improve a colluding TRA strategy in OSNs?

Our hypothesis was that we could find a path from colluders to target with a minimum of friend requests sent by using homophily and a new type of attack strategy, that addresses the trade-off between exploitation and exploration, during the infiltration process. In a colluding TRA strategy, we want to propose a new colluding attack strategy that is able to uncover a specific target in OSNs. For that, we include the principal component analysis to in our proposed information benefit to measure the closeness between two nodes.

Still, we are using homophily and topology shared knowledge between colluders to reach the target in OSNs, proposing an efficient and successful strategy remain challenging. The target(s) and their friends can have their profiles on Only Me [LSDT16] or have a very small number of friends; therefore, this can make that colluding TRA strategy not really efficient because the attack may need more friend requests to send before discovering the target.

5.3 Social Network Model

The reference case used in this work is the online social system. We model the network as an undirected graph $G = (V, E)$ that includes a set of vertices V and a set of edges E. In online network structure, a vertex $v \in V$ denotes a user and an edge $e \in E$ between two users v_i, v_j represents the friendship among them. We emphasize every user are either legitimate or malicious in the network.

5.4 Attack Scenario

In a colluding TRA, the objective of colluders C is to discover the location of a target, t, in the OSN. Upon locating t, the colluders can explore the target's friend network and extract further sensitive and private information about t to develop stronger malicious attacks such as phishing and spamming.

Every user in the OSN develops their personal profile to some extent to share information about their lives. Such profile features include the user's academic background, work experience, marital status, gender, place of birth, etc. Privacy settings in OSNs make it possible for users to control how much of this personal information and their friendships are shared with their social circles. In Facebook, for example, users can restrict access to their profile information based on three options: public, friends of friends, and Only Me [LSDT16]. We define the privacy settings in our OSN such that all $v_i \in V$, including C, have limited information about the global topology of the OSN and profile information of users. Specifically, we state that a user v_i has attained visibility of the friends of v_j if and only if $\exists$ $e_{i,j} \in E$. In other words, v_i and v_j are visible to each other if there exists a user $v_x \in V$ such that $e_{i,x} \in E$ and $e_{x,j} \in E$. Thus, if v_i befriends a user v_x, then all $v_j \mid e_{x,j} \in E$ become visible to v_i. In other words, if v_i befriends v_x, then all friends of v_x are now visible to v_i. When a user v_i attains visibility of v_j, v_i gains knowledge of v_j's profile features.

Before the attack begins, we assume that C and t are members of the same OSN, such that $C, t \in V$ and C does not have visibility of t. The colluders only have knowledge of the target's publicly available profile feature information and can take advantage of sharing their topological knowledge between them. Hence, the colluders' goal is to use this prior information to reach the target with no knowledge about the target's friends, and attain previously unattainable information upon finding the target. We emphasize that colluders at each step of their strategy will share their network topology. This may help them have a precise knowledge of user visibility in the network be useful during the infiltration strategy.

5.5 Homophily-based Similarity metrics

Weighted Similarity (WS)

In OSNs, users share features such as education, employers, birthplaces, etc. on their profiles. We congregate profile information to create a feature vector $\vec{v}$ for every user v in the OSN, where each vector index i corresponds to a specific feature characteristic, and $\vec{v_i} = 1$ if v has the characteristic, and $\vec{v_i} = 0$ if otherwise. For example, if index $i = 3$ corresponds to majoring in Cybersecurity, then every Cybersecurity major u in the OSN would have $\vec{u_i} = 1$. By representing users as vectors, mathematical methods can be applied to users.

Algorithm 3: Weighted Similarity (WS)

```
  Input  : Colluding member: $C_i$, Potential Victim: $v$
  Output: Weighted Similarity between $C_i$ and $v$: $sim_{C_i,v}$
1 $\vec{C_i} \leftarrow getUserFeatures(C_i)$
2 $\vec{v} \leftarrow getUserFeatures(v)$
3 $F \leftarrow getFriendsFeatures(C_i)$
4 $\vec{e_0}, \vec{e_1}, \vec{e_2}, \vec{e_3}, \vec{e_4} \leftarrow PCA(F)$
5 $maxSim \leftarrow \arg\max_{j \in [0,4]} cosineSim(\vec{v}, \vec{e_j})$
6 $sim_{C_i,v} \leftarrow cosineSim(\vec{C_i}, \vec{v})^{(1-maxSim)}$
7 return $sim_{C_i,v}$
```

Given each member C_i of the colluding group C and target t, Weighted Similarity (WS), defined in Algorithm 3, uses geometric distances between feature vectors to compute similarity. First, the feature vectors of each colluding member C_i and potential victim v are stored in $\vec{C_i}$ and $\vec{v}$, respectively *(lines 1-2)*. Furthermore, the feature vectors of the friends of C_i are stored in the friend feature matrix F *(line 3)*. We then perform Principle Component Analysis (PCA) on F and store the first five principle component axes in e_0 through e_4 *(line 4)*. Next, the cosine similarity between $\vec{v}$ and each axes e_i is computed and the max is stored in $maxSim$, and this would allow us to emphasize on the feature that is most related to the victim user *(line 5)*. Finally, the cosine similarity between $\vec{C_i}$ and $\vec{v}$ is computed, weighted by $1 - maxSim$, and returned *(line 6-7)*. The feature weighing would highlight the closeness between different users.

The purpose of PCA is to reduce the dimensionality of the feature matrix while identifying and weighing the most important feature characteristics of each colluding members' friends. We find that feature weighting on users can be better captured within the top five principal components. Similar results have been found by Viswanath et al. [VBC+14]. For example, consider a pair of users who majored in the same subject at the same school, and another pair of users who are the same age and gender. Intuitively, the first pair is more likely to know each other and be friends than the second pair. Without feature weighting, both pairs of users would have the same number of shared features. With feature weighting through PCA, the first pair has a higher similarity. Furthermore, PCA returns orthogonal axes which decorrelate features. For example, major and job titles are likely to be heavily correlated, i.e., Cybersecurity majors and Information Security jobs. PCA prevents overweighing these highly correlated features. Finally, PCA reduces the dimensionality of the matrix and the computational load of the algorithm.

Here we used the called cosine similarity [MBC17] to emphasize the preferential attachment between users in OSNs. It can be defined as follows:

$$Sim(x, y) = \frac{\vec{x} \cdot \vec{y}}{\parallel \vec{x} \parallel \parallel \vec{y} \parallel} \tag{5.1}$$

where $\vec{x}$ and $\vec{y}$ represent dimensional vectors of x and y, respectively.

Homophily-refined Triadic Closure (HTC)

The Triadic Closure principle [BMBR11], which states that users who share mutual connections are more likely to befriend one another, has been shown to apply to OSNs. Li et al. [LSDT16] generated the following best-fit function that predicts the probability a user v accepts a friend request from user u based on Triadic Closure:

$$accept_u(v) = p_1 * log(|M_{u,v}| + 1) + p_0 \tag{5.2}$$

where $p_1 = 0.22805837$ is the willingness of a user based on sharing mutual friends, $p_0 = 0.18014571$ is the willingness of a user to accept a new or unknown friend and $M_{u,v}$ is the set of mutual friends between u and v. Note that this would return poor results due to topology visibility limitations as mentioned previously. However, we modify this function to explore the possible benefits of utilizing homophily to improve user distinction.

Algorithm 4: Homophily-refined Triadic Closure (HTC)

```
   Input  : Colluding member: $C_i$, Potential Victim: $v$
   Output: Homophily-refined Triadic Closure metric: $MFsim_{C_i,v}$
1  $m \leftarrow getMutualFriends(C_i, v)$
2  $M \leftarrow getUserFeatures(m)$
3  $\vec{v} \leftarrow getUserFeatures(v)$
4  $r \leftarrow 0$
5  if $|m| > 0$ then
6      $s_1, s_2, s_3 \leftarrow topThree(M, WS())$
7      $r \leftarrow$mean$(s_1, s_2, s_3)$
8  end
9  $r' \leftarrow$rescale$(r)$
10 $n \leftarrow r' * |m|$
11 $p_0 \leftarrow 0.18014571$
12 $p_1 \leftarrow 0.22805837$
13 $MFsim_{C_i,v} \leftarrow p_1 * log(n + 1) + p_0$
14 return $MFsim_{C_i,v}$
```

Given each member C_i of the colluding group C and target t, Homophily-refined Triadic Closure (HTC), defined in Algorithm 2, uses homophily to modify the number of mutual friends between each colluding members C_i and potential victim v

and ultimately predict friendships. First, the mutual friends between C_i and v are stored in m, the mutual friend feature matrix is stored in M, and feature vector of the potential victim v is stored in $\vec{v}$ *(lines 1-3)*. The refinement factor r is initialized to 0 *(line 4)*. If there exists at least one mutual friend, the three greatest $WS()$ values between the mutual friend feature vectors $\vec{u} \in M$ and $\vec{v}$ are stored in s_1, s_2, and s_3 *(lines 5-6)*. The mean of these three values are saved in r *(line 7)*. The refinement factor r is then rescaled to the range $[0, 2]$ by multiplying by 2 to give r' *(line 9)*. This product r' is multiplied by the number of mutual friends, $|m|$, to give the refined number of mutual friends n *(line 10)*. This refined count n is plugged into the best-fit function and the output $MFsim_{C_i,v}$ is returned *(lines 11-14)*.

Note that due to the rescaling, r' takes values less than 1 for r values less than 0.5. Conversely, r' takes values greater than 1 for r values greater than 0.5. As a result, when r' is multiplied by the number of mutual friends, the resulting product n is either greater than the actual number of mutual friends $|m|$ if the mutual friends have high similarity (i.e., high r value), or $n < |m|$ if the mutual friends have low similarity (i.e., low r value). Thus, homophily strengthens user pairs with similar mutual friends and weakens pairs with dissimilar mutual friends.

5.6 Evaluation of Homophily-based Similarity metrics

In the following section, we explain the nature of the dataset used during simulations, outline our simulation procedures, and analyze the results as pertains to HSMs.

Standford SNAP Large Network Dataset

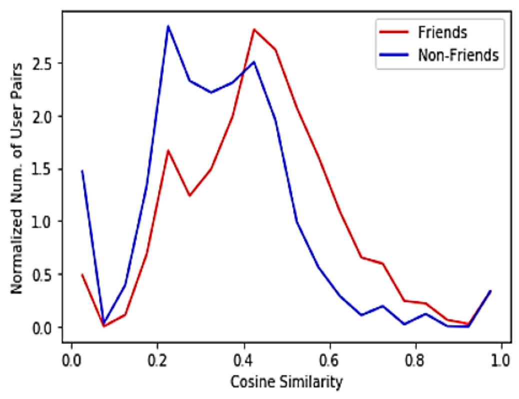
The Standford SNAP Large Network Dataset [LK14] contains data of the Facebook social graph consisting of over 4,000 nodes representing users in the social network, and roughly 88,000 edges representing friendships between these users. The dataset represents a subset of the larger Facebook, which contains 2.3 billion users. Each Facebook user updates their profile with information about their personal lives and selects what information to publicize. The possible fields users have publicly visible on their profiles include academic degree, school, graduation year, gender, hometown, employer, location of work, and academic concentration. Each user's profile information is represented by a binary feature vector as explained in Section 5.5. In summary, from this dataset we have a graph of users, users' friendships in Facebook and each user's publicly available features synthesized in a binary vector.

It should be noted that the SNAP dataset is centered around 10 ego users. The data focuses on 10 ego users, all friends of those 10 users, and the friendships among those friends. As a result, the SNAP dataset does not entirely reflect the nature of a Facebook graph due to ego users having a disproportionately large number of friends compared to other users. Thus, ego users and their incident edges were removed from the dataset so that all user friend lists would be on roughly the same scale.

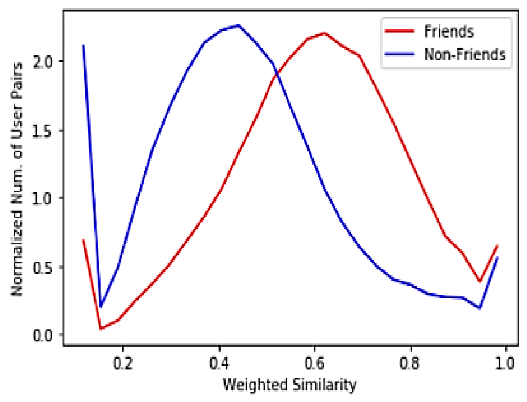
Friends vs. Non-Friends

To our knowledge, there is currently no other metrics that approximate user closeness. As such, it is impossible to compare the efficacy of our methods against an established baseline. Instead, we analyzed various properties of the metrics.

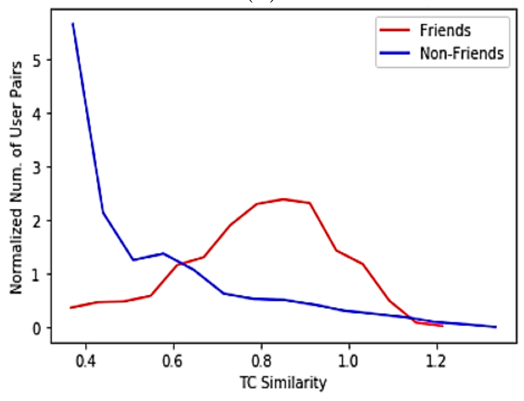
We compared the abilities of various metrics to distinguish between friend and non-friend user pairs. Out of $8,000,000$ unique user pairs, we choose $1,000,000$ to compute the metric values and separate them by scores based on friend vs. non-



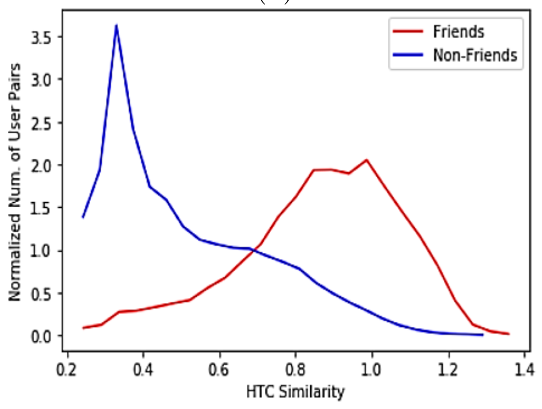
(a)



(b)



(c)



(d)

Figure 5.1: Relative frequency histograms of different metrics among friends vs non-friends.

friend users. The figures 5.1 (a-d) represent the relative frequency histograms among the friend and non-friend distributions of each metric. The following metrics were used:

- Cosine Similarity (CS)
- Weighted Similarity (WS)
- Triadic Closure best-fit (TC): See Section 5.5.
- Homophily-refined Triadic Closure (HTC)

Note that CS and WS do not rely on mutual friends to be computed while both TC and HTC do. Therefore, only user pairs with at least 1 mutual friend were used to compute both TC and HTC. This will allow for graph illegibility due to a disproportionately large number of pairs with no mutual friends can result in a score of 0.

The Kolmogorov-Smirnov 2-Sample (KS) Statistic [Lop11] was computed between the friend and non-friend sample distributions for each metric. The KS-Statistic is a measure of sample distribution dissimilarity. The results are presented in the following table:

Metric	Non-Friend Mean	Friend Mean	**KS-Statistic**
CS	0.34	0.44	**0.27**
WS	0.45	0.60	**0.36**
TC	0.53	0.79	**0.56**
HTC	0.51	0.87	**0.60**

Table 5.1: KS-Statistics of various metrics

The KS-Statistic score of the metrics in Table 1 shows that HTC best performs in distinguishing between friends and non-friends. In particular, HTC shows a 7.14% improvement over TC, highlighting the benefits of incorporating homophily. Furthermore, while both TC and HTC have higher KS-Statistic scores than CS or WS, note that these metrics are incalculable with limited topological visibility. Among the measurable metrics, WS performs 33.3% better than CS. The improvement can be seen in the greater separation of distributions in Figure 5.1-(b) compared to Figure 5.1-(a). This shows the improvements PCA provides to pure cosine similarity.

User Closeness

Since the goal of the metrics was to quantify user closeness, the average similarity based on the number of hops was also computed. Since topological metrics are restricted to only users 1-2 hops due to mutual friend reliance, only CS and WS were used. For each metric, we randomly selected 1,000 user pairs, computed their similarity scores, and labeled them based on the number of hops. For each hop number, we averaged the similarity scores and plotted the results. Note that all pairs 5+ hops were grouped together, as were all users with no connecting path. Furthermore, the WS plot has been translated down by 0.037 for easier visual comparison.

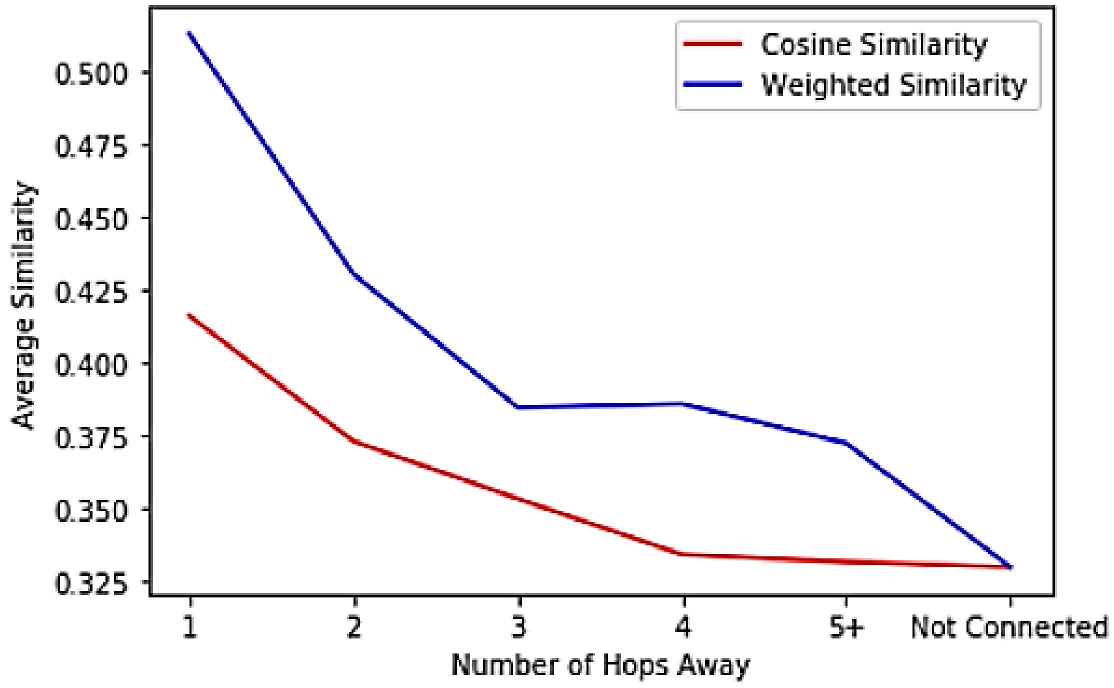


Figure 5.2: Average similarity based on number of hops away between users.

As shown in Figure 5.2, the WS metric varies more significantly as a function of user closeness. In particular, the range of mean values for CS is 0.09 while the range of mean values for WS is 0.18, showing a 100% improvement. This once again highlights the improvements provided by PCA to homophily-based metrics. In summary, based on preliminary findings, WS is best among feasible metrics for quantifying user closeness and information benefit for colluding attack in TRA.

5.7 Colluding Attack Strategy

In this section, we present the strategy utilized by colluding attackers with limited knowledge on the network topology to discover a target in an OSN.

Overview of Attack Strategy

The objective of the colluding attack strategy is to gain visibility of the target by befriending a friend of the target while sending as few friend requests as possible. Note that there are only one target and multiple victim nodes. The victim nodes will be the ones that will accept the friend requests coming from colluding members during the infiltration process. We consider the colluding attack to be successful if and only if at least one of its members gain visibility of the target node with fewer friend requests sent.

As the colluding attackers and target are already known to exist in the OSN before the attack, the colluders attain visibility of the target by sharing the network topology, gradually befriending more users in the OSN, intelligently expanding their initial set of friends until one of the colluding members befriends a target's friends. The colluding attack navigates through the social network using the homophily-based metrics discussed in Section 5.5 and explores areas of high potential benefit in the network to find users closer to the target.

The scheme of such colluding attack poses a new challenge. Precisely, when colluding members present the same behavior such as sending friend requests to the same set of users, each of the colluding members will gather almost the same information during the infiltration process. To be successful, the colluding attack should have all colluding members dynamically adapt their friend requests and send them with regard to members' network topology knowledge; this will help the colluding attack be more effective and efficient in the infiltration process.

Therefore, we propose a colluding attack that uses a two-fold strategy emphasizing exploration and exploitation characteristics to reach the target. The intuition behind this strategy is to better cope with the decision while avoiding stationary stage during the infiltration attack. The exploitation part of the strategy is derived from the metrics discussed in Section 5.5. HSMs allow each colluding member to approximate both their own closeness to the target and the closeness between potential victims and target to iteratively select who to send the next friend request. However, befriending users close to the target alone may not always result in finding a path to the target, as a group of users may be similar but not friends with the target. For example, two basketball players from opposite ends of the globe may be very similar but are unlikely to be friends in the network. Hence, we complement the exploitation strategy with an exploration part. This encourages the colluders to befriend highly visible users. The randomness and variability added by exploration help each colluding member to avoid OSN detection and prevent stagnation within one area of the network.

Algorithm Description

In Facebook and many other OSNs, however, users have the ability to accept, reject, or not to respond to friend requests. Hence, to more realistically capture the colluding attack environment, we assume that users can either accept or reject friend requests sent by any of the colluding members. Given this constraint, colluders must consider users that could reject friend requests and appropriately balance sending friend requests to users likely to accept friend requests and users likely to be near the target. The colluding attack strategy outlined in algorithm 3 considers users' acceptance behavior.

Given an OSN represented as a graph G, a colluding set C, a target t, and a maximum friend request threshold α, the colluders iteratively send friend requests in

Algorithm 5: Colluding Attack Strategy

```
Input  : Graph: $G = (V, E)$, User Set: $V$, Friendship Set: $E$,
         Colluding Set: $C$, Target: $t$, Max FR: $\alpha$
Output: If t found, number of friend requests sent: $FR$. Otherwise:
         “Target not found”.
1  $FR \leftarrow 0$
2  $CountDict \leftarrow \{\}$
3  $TopKnown \leftarrow InitialTopologyKnowledge(C, G)$
4  while $FR \leq \alpha$ do
5      for each node $C_i \in C$ do
6          Check $CountDict$
7          if $FR \mod \lambda < \frac{\lambda}{2}$ then
8              $v^{\star}_{opt} \leftarrow \arg\max_{v \in TopKnown} accept_{C_i}(v) \times WS(C_i, t)^{1-CS(v,t)}$
9          else
10             $v^{\star}_{opt} \leftarrow \arg\max_{v \in TopKnown} accept_{C_i}(v) \times VD_{C_i}(v)$
11         end
12         $FR \leftarrow FR + 1$
13         $CountDict \leftarrow CountDict \cup CountDictUser(v^{\star}_{opt})$
14         $TopKnown \leftarrow TopKnown \cup UpdateTopologyKnowledge(v^{\star}_{opt})$
15         $PV^{\star}_{C_i} \leftarrow removeVictim(PV^{\star}_{C_i}, v^{\star}_{opt})$
16         if $t \in PV^{\star}_{C_i}$ or $TopKnown$ then
17             return $FR$
18         end
19         if $FR = \alpha$ then
20             return “Target not found”
21         end
22     end
23 end
```

an effort to gain visibility of t. Before the colluding attack begins, the colluders have sent zero friend requests FR; $CountDict$ which counts the number of times potential victims have accepted or rejected the friend requests sent by each colluding member in group C is initialized as an empty set *(lines 1-2)*. Furthermore, $TopKnown$ is initialized so that all $C_i \in C$ share their initial topology knowledge between each others *(line 3)*. Until no more than α, colluding members C_i in the group C consecutively and dynamically adapt their friend requests sent to $v^{\star}_{opt} \in TopKnown$

precisely on their respective $PV^{\star}_{C_i}$ potential victims. Colluders use two strategies to ultimately increase $PV^{\star}_{C_i}$ and the colluders' topological knowledge $TopKnown$ in the OSN *(lines 4-23)*. Also, $PV^{\star}_{C_i}$ represents the set of potential victims and it is initialized as the set of each colluding members' initial friends of friends; the symbol $\star$ denotes a specific mark/characteristic to each colluding member and it will help them identify the knowledge of each of them in the network. Initially, each colluder check the rejection behavior of their potential victims *(line 6)*. The colluding attack alternates between two different strategies every $\frac{\lambda}{2}$ (λ is a parameter used by the colluders to oscillate between strategies) friend requests sent by the colluding members *(lines 7-8 or 9-11)*. After each friend request sent, the number of friend requests FR, the user's behavior $CountDict$, and the topological knowledge $TopKnown$ are updated *(lines 12-14)*. $v^{\star}_{opt}$ is removed from the potential victim list $PV^{\star}_{C_i}$ of C_i whether the friend request has been accepted or rejected *(line 15)*. However, other colluding members would keep seeing $v^{\star}_{opt}$ as a potential victim in case it appears in their potential victim list $PV^{\star}_{C_i}$. If t is in the potential victim list $PV^{\star}_{C_i}$ or in $TopKnown$ of one of the colluding members, the target has been seen and the number of friend requests sent FR is returned *(lines 16-18)*. If the number of friend requests sent has reached the threshold α and the target has not been found yet, the colluding attack terminates and *"Target not found"* is returned *(lines 19-21)*.

The following gives a detailed explanation of the two-fold strategy used by the colluding attack.

Exploitation Strategy

The exploitation part utilizes user similarity, the concept of homophily, and the network topology shares between colluding members to direct colluders to potential victims closer to the target. Specifically, since friends and non-friends in the OSN have different weighted and cosine similarity distributions as shown in Figure 2, the colluders can utilize these metrics to clearly distinguish between users that are close and distant to another user. Hence, we developed a modified version of WS for the colluders to befriend the optimal potential victim.

The modified WS computes the weighted similarity (Algorithm 5) between each colluding members C_i and target t and raises that to the value of the cosine similarity between the potential victim and target *(line 7)*. Though weighted similarity has better performance, as seen in Section 5.5, a colluding member can have the knowledge of the potential victim's friends if another colluding member has already befriended the same potential victim or one of their friends. The share network topology between colluders would help members in the group to achieve a better infiltration performance.

Exploration Strategy

The colluders explore the network during the attack by befriending users most frequently seen *(line 9)*. Each time a new victim, v, is befriended, colluders on different paths update the number of times each potential victim has been seen by including observations of users in the friend network of v. For each iteration of this exploration strategy, each colluding member selects the user with the highest view count. There is a correlation between potential victim with the highest view count and their degree in the network. Therefore, by befriending this type of potential victim in the exploration strategy, colluding members would rapidly increase their topological knowledge.

During our two alternating strategies explained in algorithm 5, we want to emphasize that each colluding member C_i selects a user from their set of potential victims $PV^{\star}_{C_i}$ who has the highest expected utility. We calculate the expected utility by taking the product of the probability that a potential victim v accepts C_i's friend request and the utility of that potential victim *(lines 7 & 9)*. The function $accept_{C_i}(v)$, defined in section 5.5, provides the probability that v accepts the friend request of C_i and is based on the Triadic Closure Principle [LSDT16] which measures the likelihood of a friendship between two users.

At the beginning of the colluding attack, colluding members would share between them the network topology to have the exact location and knowledge of others in the network. Moreover, this would prevent them from the same behavior by befriending the same set of potential victims and gathering almost the same information. Because of the trade-off between the two-fold strategy, colluders must have better information about the network to facilitate their infiltration with few friend requests to uncover at least a target's friend. The algorithm 6 gives a detailed explanation of the initial network topology shares between members of the colluding attack.

```
Algorithm 6: InitialTopologyKnowledge($C, G$)
   Input  : Colluding Set: $C$, Graph: $G_{C_i} = (V, E)$
   Output: Initial Topological Knowledge of each colluding members:
           $InitTopKnow_{Ci}$
 1 $InitTopKnownShare \leftarrow 0$
 2 for $C_i \in C$ do
 3     $InitTopKnown_{Ci} \leftarrow 0$
 4     $PV^{\star}_{C_i} \leftarrow getPotentialVictims(C_i, G_{C_i})$
 5     $VD^{\star}_{C_i} \leftarrow UpdateVictimViews(PV^{\star}_{C_i})$
 6     $InitTopKnown_{Ci} \leftarrow InitTopKnown_{Ci} \cup \{PV^{\star}_{C_i}, VD^{\star}_{C_i}\}$
 7 end
 8 for $i \leftarrow 1$ to $n$ do
 9     $InitTopKnownShare \leftarrow InitTopKnownShare \cup InitTopKnown_{Ci}$
10 end
11 for $C_i \in C$ do
12     $InitTopKnown_{Ci} \leftarrow InitTopKnownShare$
13 end
14 return $InitTopKnown_{Ci}$
```

Algorithm 6 takes as input a set of colluders C and the sub-graph of each of them G_{C_i} to return an initial topological knowledge shared $InitTopKnown_{Ci}$ between members of the colluding attack. The colluders have not yet shared their network topology, $InitTopKnownShare$, and it is initialized to zero *(line 1)*. For each member of the colluding group C_i, we set the initial network topology knowledge of each member to zero, $InitTopKnown_{Ci}$, their potential victims, $PV^{\star}_{C_i}$, as the set of each colluders' initial friend of friends; also, $VD^{\star}_{C_i}$, which counts the number of time a user has been seen by each colluding members is initialized; moreover, each colluding members' $PV^{\star}_{C_i}$ and $VD^{\star}_{C_i}$ are updated to a common initial topological knowledge set $InitTopKnown_{Ci}$ *(lines 2-7)*. Furthermore, the knowledge of all colluding members is updated in $InitTopKnownShare$ *(line 8-10)*. Finally, the topological knowledge of colluding members is shared and returned as $InitTopKnown_{Ci}$ *(lines 11-14)*.

```
Algorithm 7: UpdateVictimViews($PV^{\star}_{C_i}$)
  Input  : Colluding members: $C_i$, Newly Befriended Victim: $v$,
           Potential Victim List: $PV^{\star}_{C_i}$, Potential Victim View Count
           Dictionary: $VD^{\star}_{C_i}$
  Output: Updated Potential Victim View Count Dictionary: $VD^{\star}_{C_i}$
1  $F_{C_i} \leftarrow getUserFriends(C_i)$
2  $F_v \leftarrow getUserFriends(v)$
3  for $f \in F_v \cup PV^{\star}_{C_i}$ do
4      $VD^{\star}_{C_i}\{f\} \leftarrow VD^{\star}_{C_i}\{f\} + 1$
5  end
6  for $f \in F_v \cap F_{C_i}$ do
7      for $f' \in getFriends(f) \cap PV^{\star}_{C_i}$ do
8          $VD^{\star}_{C_i}\{f'\} \leftarrow VD^{\star}_{C_i}\{f'\} + 1$
9      end
10 end
11 $VD^{\star}_{C_i} \leftarrow VD^{\star}_{C_i}.remove(v)$
12 return $VD^{\star}_{C_i}$
```

The potential victim view count algorithm is depicted in Algorithm 7. The friends of each colluding member C_i and befriended victim v are stored in F_{Ci} and F_v, respectively *(lines 1-2)*. For every user that is a potential victim or friends of v, we increment their view count by 1 *(lines 3-5)*. For every user f that is a friend of both C_i and v, we also increment the view count of the friends of f by 1 *(lines 6-10)*. Once a potential victim is befriended, the user is removed from the dictionary *(line 11)*. Finally, the updated view count dictionary of each colluding member $VD^{\star}_{C_i}$ is returned *(line 12)*.

By keeping track of the number of times each colluding member has seen each potential victim, the colluders have the ability to prioritize highly connected users. This provides the colluding attack with a sense of memory, enabling a counter that keeps track of the number of times each victim's friend is seen by different members of the colluding group. It allows the colluders to have an idea of who is more frequently being seen nearby during the infiltration process. By targeting highly visible users, the colluders can discover as many users as possible per friend request, offering new routes to explore while using as few friend requests as possible.

```
Algorithm 8: CountDictUser($C_i, v^{\star}_{opt}$)
  Input  : Colluding members: $C_i$, Newly Sent Friend Request: $v^{\star}_{opt}$
  Output: Updated Potential Victim Behavior on Friend Requests
          Received: $CountDict$
1  $CountDictAccept\{v^{\star}_{opt}\} \leftarrow \{\}$
2  $CountDictReject\{v^{\star}_{opt}\} \leftarrow \{\}$
3  for $\forall v^{\star}_{opt} \in V$ do
4      if $v^{\star}_{opt}$ accepts $FR$ from $C_i$ then
5          mark $v^{\star}_{opt}$ as $v^{\star +}_{opt}$
6          $CountDictAccept\{v^{\star}_{opt}\} \leftarrow CountDictAccept\{v^{\star}_{opt}\} + 1$
7      else
8          mark $v^{\star}_{opt}$ as $v^{\star -}_{opt}$
9          $CountDictReject\{v^{\star}_{opt}\} \leftarrow CountDictReject\{v^{\star}_{opt}\} + 1$
10     end
11 end
12 $CountDict \leftarrow CountDictAccept\{v^{\star}_{opt}\} \cup CountDictReject\{v^{\star}_{opt}\}$
13 return $CountDict$
```

Algorithm 8 returns an update on users' behavior for friend requests sent by each colluding member during the infiltration strategy. With this algorithm, the members of the colluding attack can have knowledge of each potential victim's behavior. This would help colluding members during the infiltration attack to avoid sending friend requests to potential victims with high rejection behavior, therefore minimizing their number of friend requests to reach a target's friend.

Given each colluding member C_i and a new friend request sent to a potential victim $v^{\star}_{opt}$, Algorithm 8 iteratively counts the number of times each potential victim accepted or rejected friend requests coming from each colluding member. Before the colluding attack begins, colluding members have not yet sent a friend request, and the accept and reject count is initialized as zero *(lines 1-2)*. For all friend requests sent to potential victims during the infiltration strategy by each colluding member, the algorithm counts how many have been accepted or rejected by which members of the colluding group *(lines 3-11)*. Finally, the count number of accepted or rejected friend requests, sent by each colluding member, is updated and returned *(lines 12-13)*. This algorithm allows colluders to keep track which members had been rejected or accepted by potential victims; therefore, it will help them to avoid sending friend requests to this type of user even if they are their $v^{\star}_{opt}$ at that moment. For instance, if a potential victim has the tendency to reject a lot of friend requests and colluders are aware of that, then they can rather send their friend requests to potential victims that present the second highest expected utility. Therefore, the awareness of such users' behavior by colluders will help them for an effective and efficient colluding attack.

The topological knowledge described in Algorithm 9 differs from the one outlined in Algorithm 6 by considering the potential victim's behavior on each friend request sent. Potential victims can accept or reject coming friend requests. If they accept friend requests, the information on the network topology between colluders needs to be updated for an efficient and effective strategy. We emphasize that Algorithm 6

```
Algorithm 9: UpdateTopologyKnowledge($C_i, v^{\star}_{opt}$)
Input  : Colluding Set: $C$, Potential Victims List: $PV^{\star}_{C_i}$, Potential
         Victim View Count Dictionary: $VD^{\star}_{C_i}$, Graph: $G = (V, E)$
Output: Update Topological Knowledge of each colluding members:
         $TopKnown$
1 for $C_i \in C$ do
2     while $v^{\star}_{opt} == v^{\star+}_{opt}$ do
3         $PV^{\star}_{C_i} \leftarrow updatePotentialVictims(C_i, v^{\star+}_{opt})$
4         $VD^{\star}_{C_i} \leftarrow UpdateVictimViews(C_i, v^{\star+}_{opt}, PV^{\star}_{C_i}, VD^{\star}_{C_i},$
5           $CountDict)$
6         $TopKnown \leftarrow TopKnown \cup \{PV^{\star}_{C_i}, VD^{\star}_{C_i}, CountDict\}$
7     end
8 end
9 return $TopKnown$
```

just allowed colluding members to share the initial network topology between them before the colluding attack begins.

For every member C_i in the colluding group C, if the potential victim with the maximum expected utility accepts the friend request of a colluding member, we have to update the set of potential victims $PV^{\star}_{C_i}$ of that colluding member C_i; also, we have to update the number of times a colluding member $VD^{\star}_{C_i}$ has seen users in the new information gained by the friend request accepted. We emphasize that a user can be seen at multiple times by different colluders, which help colluders apply the infiltration strategy. Furthermore, the topological knowledge is updated based on newly provided information *(lines 1-8)*. Finally, the algorithm returns the new topological knowledge to the colluding member. This algorithm gives a general idea to each colluding member the position of others in the network.

To summarize this colluding attack strategy, given a more realistic OSN environment, the expected utility measurement and the shared topological knowledge allow colluding members to send friend requests to users that are both willing to accept the friend request and lead to the target. It is important for the colluding attack strategy to acknowledge two main factors. First, colluders should avoid only sending friend requests to users near the target that are unlikely to accept them; otherwise, they will end up with a large number of friend requests sent without propagating in the network. Second, colluders should avoid only sending friend requests to users likely to accept them, which will not efficiently direct them towards the target.

5.8 Experimental Evaluation and Analysis

In the following section, we would like to evaluate the effectiveness and efficiency of our proposed colluding attack strategy through experiments. We first explain the outline of our simulation procedure, then we outline the comparing strategies and provide an analysis of results.

Simulation Procedure

For our simulations, we used the Facebook SNAP dataset explained in Section 5.6. Note that the removal of ego users and their incident edges mentioned in Section 5.6 ensures ego users could not provide disproportionately large network visibility compared to other users. To test the attack strategy, we ran 2000 colluding attack trials, where an attack is defined as the termination of Algorithm 5. For each trial, each colluding member, C_i, and target, t, are users chosen randomly and independently in the social network. Precisely, we only use one/same target for the colluding attack. In addition, to run our experiments, we choose a random number in (0, 1] as the probability of any user to accept a friend request during the simulations [ND16, LSDT16]. This was done because the data do not include any further information to compute the needed probabilities.

According to Edunov et al. [EDF+], all Facebook users are on average 3.57 friend links away from each other. Hence, we ran simulations in scenarios where all the randomly selected colluding members (10) are located at minimum either four or five hops away from the target. For instance, if users a and t are four hops away, then the following edges exist in the social network graph for some users x, y, z: $e_{a,x}$, $e_{x,y}$, $e_{y,z}$ and $e_{z,t}$. Furthermore, if users a and t are five hops away, then the following edges exist in the social network graph for some users w, x, y, z: $e_{a,w}$, $e_{w,x}$, $e_{x,y}$, $e_{y,z}$ and $e_{z,t}$. In other words, if a and t are four hops away from each other, a needs to befriend at least two users to attain visibility of t, and if a and t are five hops away, a needs to only befriend a minimum of three users to attain visibility of t.

We were concerned with how quickly the colluding attack strategy gained visibility of the target, so we recorded the number of friend requests sent for each attack. A maximum of 200 friend requests was allowed to reach the target ($\alpha = 200$). Each colluding member alternated between strategies every 5 friend requests sent ($\lambda = 10$). We emphasize the values of each the parameters (α, C, and λ) mentioned above depend on the colluders willingness to uncover the target without been detected. It is important to note that there is a linear correlation between the number of colluders and the severity of the attack. Studies [LL16, LL18] show that the higher the number of colluding members, the better the attack success rate would be.

Comparing Strategies

We compare our colluding attack strategy outlined in Algorithm 5 (CTRA) with other iterative victim selection strategies. Our proposed CTRA is the only targeted attack strategy that utilizes profile feature information of users in an OSN to reach the target and calculable with restricted topological visibility. We have indicated earlier the restrictions on [ND16, LSDT16, LST17] for using only infiltration metric that rely on network topology, neither consider the colluding scenario or include profile features; or do not propose an infiltration strategy and assume to have the knowledge on the target network topology [LL16, LL18]. For those reasons, we are not considering them for comparison. We instead compared our CTRA strategy against victim selection strategies. They are based on metrics that are iteratively used to befriend potential victims based on the optimality of their topological features. We compare our CTRA to the following victim selection strategies:

Random strategy: selects a potential victim at random from among all the possible potential victims that the colluding member has knowledge on. This strategy has the advantage of requiring minimal information, i.e., only what potential victim the colluding member can see at the moment. The probability of a potential victim x being selected to send a friend request is as follows:

$$p(x) = \frac{1}{|PV^{\star}_{C_i}|} \tag{5.3}$$

Jaccard Coefficient [LFS17]: selects the node with the maximum number of mutual friends with a colluding member, normalized by the total unique friends of both users. The probability of a potential victim x being chosen to send a friend request is as follows:

$$p(x) = \arg \max_{u \in PV^{\star}_{C_i}} \frac{|N(C_i) \cap N(u)|}{|N(C_i) \cup N(u)|} \tag{5.4}$$

where $N(.)$ denotes the neighbor.

Maximum Degree: selects a potential victim that has more friends from among all the possible potential victims that the colluding member has knowledge on. The probability of a potential victim x being selected to send a friend request is as follows:

$$p(x) = \arg\max_{u \in PV^{\star}_{C_i}} \frac{|N(u)|}{\sum\limits_{v \in PV^{\star}_{C_i}} |N(v)|} \tag{5.5}$$

Triadic Closure [LSDT16]: selects a potential victim that has the maximum number of mutual friends with the colluding member (see Section 5.5).

Random selection serves as a control. Jaccard and Triadic Closure were chosen due to their ability to predict friendship. Maximum degree was chosen as a purely exploratory metric.

Performance Evaluation

Our objective is to show the effectiveness of CTRA strategy in gaining visibility of the target while using as few friend requests as possible. In addition, we compare the difference in performance of our CTRA in colluding attack environments. We also present a single attack using our algorithm 5. Furthermore, we emphasize the difference between a colluding and a single attack in a network. Note that for clarity of our simulation tests, we use the empirical cumulative distribution function to capture attacks that have been completed in different friend requests threshold.

The comparison between the colluding attack strategies is illustrated in Figure 5.3. It is remarkable to see how our CTRA presents a higher percentage of the attack trials that attain visibility of the target by sending fewer friend requests. CTRA significantly and steadily outperforms the other strategies for any values of friend requests. In addition, 94 % of the attack trials are completed within 40 friend requests while the highest of the other strategy (TC) reach 63.5%. The Triadic Closure and the Jaccard have similar behavior regarding the selection strategy; however, the TC presents the advantage of just focusing on befriending potential victim with

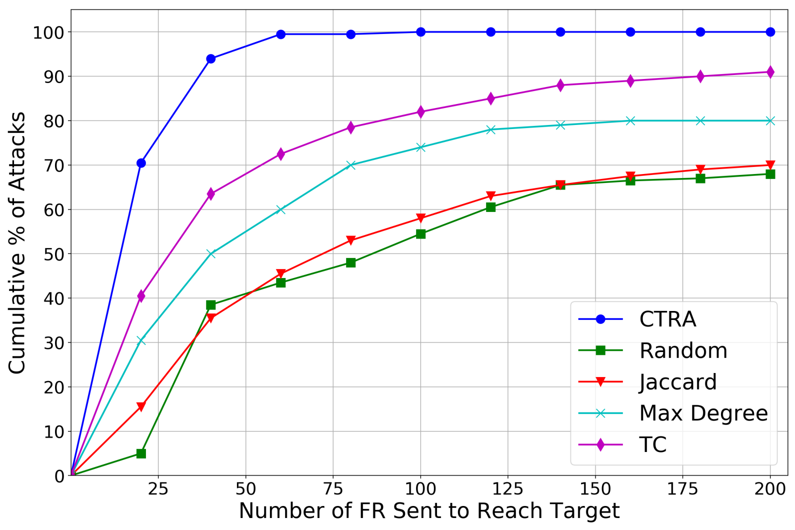


Figure 5.3: Comparison between different attacks strategies.

a maximum number of mutual friends with colluders and not on the normalized mutual friends as Jaccard. Even though the Max Degree strategy performs better than Jaccard, we can infer that MD found users who could lead to the target. In contrast, the Random strategy performs the worst; this could be explained by the strategy itself. The Random does not choose a potential victim based on any type of strategy; therefore, it can get a lot of rejection during the infiltration strategy. The results obtained by our CTRA is coherent due to our selection strategy of the next potential victim with the greatest expected information benefit. In addition, colluders will avoid sending friend requests to the users that have a reputation of rejecting friend requests. Furthermore, colluders would share their knowledge and exploit different paths in the network in the aim to avoid having the same information benefit; this case can happen if colluders are exploiting/exploring the same sub-network. Hence, we demonstrate the intuitive notion that features, homophily, and selection strategy matter in any type of infiltration procedure to attain the visibility of a target.

We also want to investigate the case of a single attack in targeted reconnaissance attack (TRA) against the other selection strategies. It would allow us to verify that our strategy can perform and present quality results in both cases (colluding and single attack). Strategies are also compared based on the percentage of attacks trials that are completed within a certain number of friend requests. Note that TRA strategy result from CTRA but just one attacker is involved; therefore, sharing knowledge for better exploitation/exploration is not allowed.

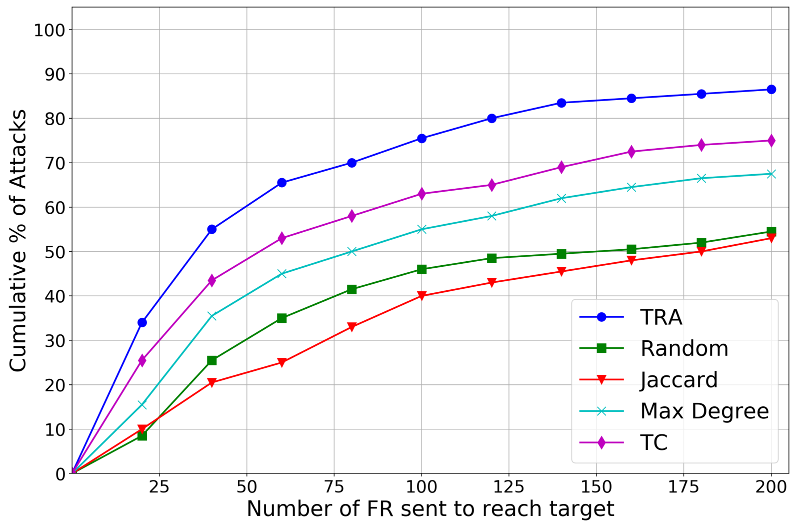


Figure 5.4: Performance of different single attack strategies to attain visibility of target.

The next observation from Figure 5.4 shows the number of friend requests used by each strategy to attain a given target in the network. From this, we observe that our TRA strategy outperforms other selection strategies. The difference in performance is considerably significant. In a single attack, no strategy was able to reach the 100% attack trials within the 200 friend requests. We can infer this happens because the attacker or the target is disconnected in the network. Furthermore, the further the target, the less TRA would better perform. Another clear difference is that Random strategy performs better than the Jaccard even though potential victims are randomly selected without any examination of information gain. For this case, we can infer the random chose potential victims with better information benefit during the infiltration strategy. Also, Figure 5.4 shows that our TRA gets the most information benefit from befriending potential victims within the target friends to succeed. Furthermore, TC and MD strategies do not befriend as much as TRA around the target, they perform similarly and lack the effectiveness of the TRA. The poor performance of other strategies can be explained as the first potential victims they befriended have a better information benefit at the beginning of each strategy. However, as they progress, the information benefit decreases when the number of friend requests gets higher. This is comprehensible due to the selection process of each strategy which chooses the potential victim with the highest value/gain.

The last observation from Figure 5.5 emphasizes how a colluding attack can outperform a single attack while both are using the same strategy.

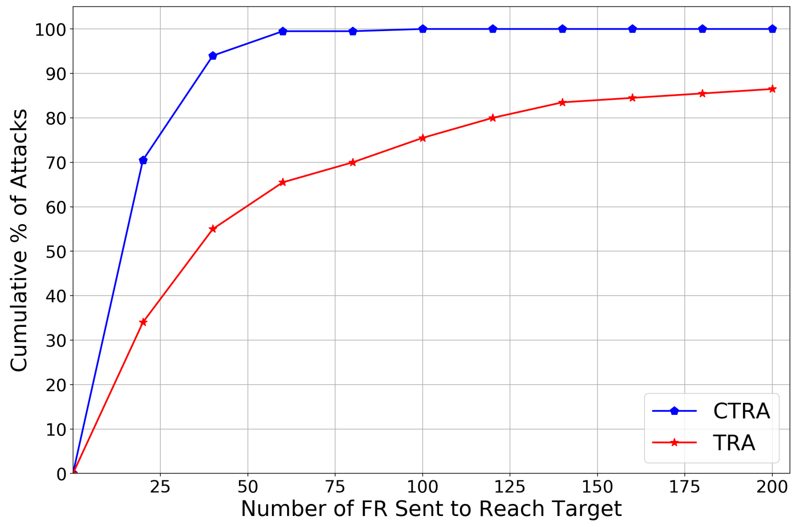


Figure 5.5: Performance of different single attack strategies to attain visibility of target.

It can be seen using a colluding attack strategy with a maximum threshold of 200 friend requests, malicious users involved in the attack utilize fewer friend requests to attain visibility of the target with a higher percentage of the attack trials. 94% of the attack trials are completed within 40 friend requests using CTRA while TRA would never reach that success. Since a user can have a large number of users at each hop to select and befriend in OSNs; a single attacker strategy would not be able to consistently performs better than a colluding attack. Furthermore, we want to point out that each attacker involves in different strategy do not have the entire knowledge of the network but just the users that are 2 hops from them. Therefore, an attack that involves a collaboration between malicious users would provide a significant advantage to each malicious users such as a node behavior, sharing the network topology to better exploit/explore the network.

We show that utilizing target profile information, such as features, can help in attaining target visibility. While other methodologies focus on simply propagating in an OSN using victim selection strategies, we demonstrate the necessity of intelligently utilizing targets' profile information to direct the colluders towards their targets more effectively. When designing an attack strategy, it is crucial to utilize information about the users' features rather than just the topological information of the OSN, such as the users number of total friends or number of mutual friends. Thus, the concept of homophily can be used to guide more effective attacks.

5.9 Summary

In this chapter, we present a new colluding attack pattern to infiltrate a network without being detected by utilizing a minimum number of friend requests. This aim to show the vulnerabilities users may encounter by targeted reconnaissance attacks in OSN. Malicious users involve in colluding targeted reconnaissance attack (CTRA) do not have the knowledge about the network topology, even though they want to attain the visibility of a specific target. In contrast to other methods [ND16, LSDT16, LST17] that use network topology to develop infiltration strategy, we propose an information benefit metric to approximate user closeness in an OSN through the use of homophily [KML$^+$19]. This information benefit metric would help direct the colluders in TRA towards a specific target. Specifically, we develop and present the advantage of our Weighted Similarity that utilizes users profile features and PCA to approximate closeness. We do this by introducing an attack (CTRA) that uses a two-fold strategy to attain the visibility of the target. From our results of the infiltration process on real social network dataset, we have seen that the sociological theory (homophily), behind friendship formation in the network, plays a vital role in the success or failure of the infiltration process. Finally, our experiments demonstrate the benefits of using homophily, as well as the promise of the CTRA strategy. Future direction and limitations will be given in Chapter 7.

CHAPTER 6

COMBATING COLLUDING RECONNAISSANCE ATTACK IN ONLINE SOCIAL NETWORKS

In chapter 4, we investigate the detection of colluding ICA and found that colluders could be easily detected because of their main characteristic of building new social circles to lure the target. Therefore, detecting one colluder will reveal the others because of their connectivity. In addition, colluding ICA know the topology of the network. In contrast, colluding targeted reconnaissance attack (TRA), attackers build their own legitimacy by sending friend requests to users that share some similarities (e.g., number of mutual friends, attribute similarities); therefore, making it more suitable for the genuine OSN users to accept their friend requests. In addition, colluding targeted reconnaissance attacks do not build isolated groups of friends and do not connect with other attackers (i.e., sending friend requests to each other. Therefore, we develop a novel community detection method under a framework of node-centric approach (i.e., not considering the network as a whole). Our proposed method integrates network topology and user behavior, and introduces novel metrics to better identify colluders between friend requests sent to a target user in the network. The outline of this chapter is as follow: an introduction of this problem will be given in Section 6.1. The problem statement will be defined in Section 6.2. In Section 6.3, the social network model will be described. After that, the detailed threat model is presented in Section 6.4. The design and the detection of ego-centered network method will be introduced and evaluated in Section 6.5. The proposed approach for the colluding group in TRA and the detailed solution will be presented in Section 6.6. In Section 6.7, experimental results will be presented to evaluate the performance of our proposed methods. Finally, we give a short summary of this chapter in Section 6.8.

6.1 Introduction

In literature, many attacks have been discussed regarding OSNs [RSL$^+$17]. In particular, one of the malicious activities performed by attackers in OSNs, called friend spam [SCM11], consists of sending multiple friend requests to users in OSNs even though they are not related to that user. The attackers aim to have as many users accept their friend requests. However, the price of accepting these friend requests by unaware users is very high. For instance, the opinion manipulation trolls behavior across communities discussions [CGH18], [KCLS17]. Even though many works have been proposed to cope with malicious friend spam [SCM11], [CSYM15], one novel malicious behavior called reconnaissance attack (RA) has succeeded in evading defense mechanisms for friend spam [ND16], [FGE14]. In an RA, an attacker sends friend requests to users that share some similarities such as the number of mutual friends and attribute similarities, therefore making more suitable for the genuine OSN users to accept their friend requests [MSLC01]. An extension of the RA is the targeted reconnaissance attack (TRA), in which the attacker wants to befriend a specific set of victims users to reach the target. This type of attack (i.e., TRA) will increase the probability of friend requests sent to be accepted by the target [ND16]. Specifically, Sybil defense mechanisms [BBR13] will fail to detect such type of malicious behavior. This chapter aims to address the problem of detecting collusion in TRA. Precisely, we monitor the friend request sent by any users to specific targets that we define and detect as malicious the corresponding ones. To be resilient to the collusion behavior observed in OSNs, we formulated the colluding malicious friend requests as ego-centered networks and overlapping community detection problem.

6.2 Problem Statement

Since OSNs providers developed their foundation on social graphs and generally assume that a social network only consists of friendships symbolizing social trust between a pair of users, accepting a malicious friend request can be disastrous [CGH18]. These attacks waste network resources, take up resources from the social search engine, allow for viral marketing, circumvent OSNs defense techniques, and violate users' shared content. Therefore, we want to give a response to the next question:

Can an ego-centered network model that includes both graph-based (topology) and behavior-based (communication) characteristics mitigate colluding TRA in OSNs?

We assume that the social graph connections among colluders around their target would be different than legitimate users. Similar assumption has been made by Wang et al. [WZG18] as well. As a result, all colluders' friends from friendship relations between each other would present an inconsistency of the graph. The objective is to classify friend requests received by target(s) as colluders or legitimate. To mitigate colluding TRA, we aim to propose a defense mechanism that would also include novel metrics to cope with colluders behavior.

Despite using both topology and behavior characteristics of each user, introducing an effective approach constitutes a new challenge. Colluders in TRA can avoid creating an isolated group of friends and participate more each social circle they are involved with; therefore, this can make the detection mechanism less efficient.

6.3 Social Network Model

We represent an OSN as an directed graph $G = (V, E, w)$, where V is the set of users and E is the direct connection between two users $v_i, v_j \in V$ represented as

$e_{i,j} = (v_i, v_j) \in E$. w represents the number of social interaction on each directed edge. User interactions differ from the type of OSN, Facebook (e.g., messages, photo uploads, etc.), LinkedIn (professional acquaintance) [OSH+07, VMCG09b].

In this work, we particularly emphasize on undirected social networks as explained in Section 6.7. Based on each ego node (EN) describes in the section 6.5, they are two types of relationships such as dyadic and triadic. The dyadic relationship (e.g., social tie) represents the direct connections between the EN and its neighbors (i.e., friends) while triadic relationship (e.g., social triad) represents the connections between its neighbors (i.e., friends). A social triad better represents the social relationships of the ego-centered network (ECN) of an EN.

6.4 Threat Model

Colluders in TRA have the objective that at least one of the attacker $a \in A$ in their malicious group will have their friend request accepted by their common target, t, in the OSN. For instance, the target t can be users working at companies that have information needed by attackers [PPS14, PSP+17]. Upon a friend request accepted by t, colluders are only interested on sensitive information shared between t their friends to further sell or use them [FGE14, BLM+17], they are not engaged of having the target profile information.

Every user in the OSN develops their personal profile to some extent to share information about their lives. Such profile features include the user's academic background, work experience, marital status, gender, place of birth, etc. Privacy settings in OSNs make it possible for users to control how much of this personal information and their friendships are shared with their social circles. In Facebook, for example, users can restrict access to their profile information based on three options: public, friends of friends, and Only Me [LSDT16]. Specifically, we state that a user v_i has attained visibility of the friends of v_j if and only if $\exists\ e_{i,j} \in E$. In other words, v_i v_j are visible to each other if there exists a user $v_x \in V$ such that $e_{i,x} \in E$ and $e_{x,j} \in E$. Thus, if v_i befriends a user v_x, then all $v_j \mid e_{x,j} \in E$ become visible to v_i. In other words, if v_i befriends v_x, then all friends of v_x are now visible to v_i. When a user v_i attains visibility of v_j, v_i gains knowledge of v_j's profile features.

To avoid having rejection by sending direct friend requests to the target [BBR13, FGE14, CSYM15], colluders may try to have mutual friends with the target to increase their chance of acceptance [LSDT16]. Therefore, we assume each member a of the colluding group A is already friend with some friend of friend of the target user t such as $|MF(a,t)| \geq 1$, where $|MF(a,t)|$ represents the set of mutual friends between a and t.

6.5 Design and Proposed Method for Detection of Ego-Centered Network

In this section, we first give some definitions to clarify the proposed approach. After, we assume an ego-centered network as first proposed by [New03] and then justify why this best fit our network. Later, we describe our objective function. Furthermore, we provide an ego-centered network detection method along with a selection process of the nodes to be added to form the ego-centered network.

Definitions

We present some definitions that we will use throughout the article to make it easier to understand the use of each of them. In addition, we use vertex, node, and user for the same meaning in this article. Further, we emphasize the nodes u and v belong to a set of nodes V in the networks, i.e., $u, v \in V$ (Section 3.1).

The friend list (FL) of any node can be defined as the set of nodes that are directly connected (or 1 hop away) to a given node and can be expressed as follows:

$$FL(u) = \{v : (u, v) \in E\} \tag{6.1}$$

In addition, the mutual friend (MF) of any two nodes represents the set of common nodes that two given nodes share in the network. It can be determined as follows:

$$MF(u, v) = (FL(u) \cap FL(v)) \tag{6.2}$$

where u,v$\in V$.

Also, MF and FL can be defined for a set of nodes $C \subseteq V$ and $u, v \in V$ respectively in the following form:

$$MF(C, u) = \cup_{v \in C} MF(v, u) \tag{6.3}$$

$$FL(C) = \cup_{v \in C} FL(v) \tag{6.4}$$

The weight that can be found on the edge/friendship between two nodes u and v in the network is denoted by $w(u, v)$ and quantifies the level of communication (Section 6.3), we provide the following definitions for w_C^{in} and w_C^{out}:

- $w_C^{in} = \sum_{u \in C} w(ego, u)$ is the sum of friendship weights between ego node and other nodes inside the community C.

- $w_C^{out} = \sum_{u \in \overline{\mathrm{C}}} \sum_{v \in C} w(u,v)$ represents the sum of friendship weights that a node (v) inside the community C has with others outside C ($u \in \overline{\mathrm{C}}$) with $(u,v) \in E$.

Finally, ECN_i represents the ego-centered network formed around the user i.

The following subsection will give an overview of the ego-centered network (ECN) and its importance in our proposed detection mechanism.

Ego-Centered Network

It has been shown in Lancichinetti et al. [LF09] that weighted edges represent significant information in graph's properties and failure to consider may result in limitation of features needed for communities detection algorithms. We choose the node-centric approach to cope with the user's behavior and topology that can be revealed to a specific target in real-world OSNs. This approach would allow us to create a community formed around each node that sends a friend request. For instance, the community will be built around the user that sends a friend request. The user's friends will be tested to decide whether to be added to the community. An ego-centered network can be defined as follows:

An *Ego-centered network* can be defined as the sets of nodes that are within a given range from a particular node called the "ego" [New03].

We want to emphasize the main differences between our proposed ego-centered network and the one proposed by [MS17]. The first difference is the method of adding a new node in the community. Moctar et al. [MS17] added a new node by minimizing their quality function value based on a predefined threshold. Here, the candidate node to be added to the new community needs to have a quality function value that will continue to decrease until the overall quality function value reaches that threshold. Unfortunately, this method would not usually form a good community around the ego node because the quality function value can be higher than the threshold. Therefore, this forces the method to have a customized threshold for each node that would like to have its ECN. In other words, using their method, we cannot have a global threshold. Second, the updated process in their method adds a new node based on the ego node rather than the newly formed community. Hence, this updating process would not explicitly represent the real community built around the ego node. Usually, the candidate node to be added needs to be compared to the other existing nodes in the community of the ego node; this would unveil the interaction between the nodes in the new community. With these limitations, the proposed approach of [MS17] is not appropriate for real-world implementation. These elements enforce us to introduce an entirely different objective function and similarity-based behavior structure metric to accurately find the real community based on the friend list of any node that sent friend requests.

Our proposed ego-centered network overcomes the limitations mentioned above and the following definition explains and describes how communities are built.

Ego-centered network (ECN) can have a different range. An ECN with a range 1 includes everyone within 1 hop of friendship from the ego node (EN), while an ECN with a range 2 contains friends of friends of the EN. We show an illustration of an ECN with range 2 in Figure 6.1.

As illustrated in Figure 6.1, the green node represents the ego-node. The yellow nodes represent the direct friends of the EN within a range 1. The red nodes represent the friends of the friends of the EN within a range 2.

Objective Function

The following defined equations (6.5 - 6.8) came from modified formulas of [MS17]. We introduce an objective function that analyzes the social cohesion existing between the node that sends the friend request (FR) and their friend within 1

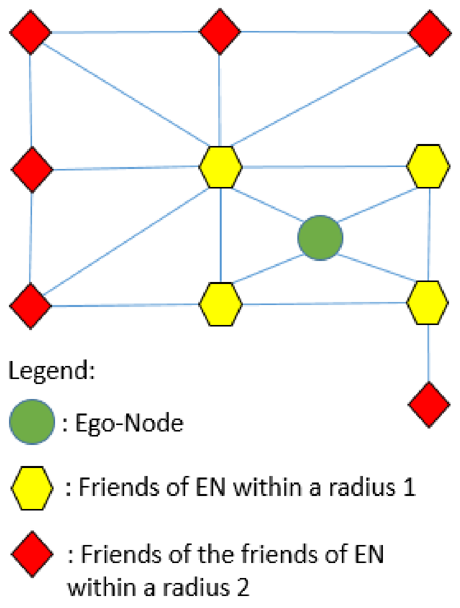

Legend:
: Ego-Node
: Friends of EN within a radius 1
: Friends of the friends of EN within a radius 2

Figure 6.1: An ego-centered network with radius 2 in OSN.

hop. This function builds the ECN by using both graph-based and behavior-based factors. Precisely, our objective function is composed of two community parameters as follows:

Behavior-based factor. This factor considers weight/communication (e.g., emails, scientific collaborations, likes, posts) between nodes inside the community (C) and nodes outside it.

$$BF_C = \frac{w_C^{out}}{w_C^{in}} \tag{6.5}$$

where w_C^{in} and w_C^{out} are defined above (Section 6.5).

Sparsity-based factor. The factor considers the community structure formed by the nodes that belong to C. It represents how cohesive the nodes are inside the community C.

$$SF_C = \frac{|V_C|}{|E_C|} \tag{6.6}$$

where $|V_C|$ and $|E_C|$ represent the number of nodes in C and the number of friendships among them respectively.

Therefore, the lower the BF_C, the better the community is partitioned in the entire graph. Also, a similar characteristic is observed when SF_C is lower, and it shows how close/cohesive is the community in regard to the network topology.

As a result, to have a better representation of each user's ECN who sent an FR to the target(s), we consider both factors to define our objective function and denote it as follows:

$$\mathcal{Q}(C) = B_f \times C_f \tag{6.7}$$

However, this quantitative measure can have a zero score if one of the factors is zero. The above-mentioned problem only occurs in one of the following cases as follows: First, The community, formed by a user who sent the FR, is partitioned (i.e., all the user's connections belong to the community), i.e. $\sum_{i=1}^{n} w_C^{out}(i) = 0$. Second, The community is formed by a single user without any friend. This case will not happen because all the nodes that send friend requests have neighbors (friend list) and by definition community is formed by at least two users (the ego-node and at least one of their neighbors).

Ego-Centered Network Detection Method

Algorithm 10 is used to evaluate the ego network formed around each node that send the friend request (ego node) to a specific target. OSNs users always post, comment on their posts, exchange message, etc. [EFKE13, PSP15, PSP^{+}17] show that malicious users in target cyberattacks do not participate in the social circle of their friends. Therefore, by building the ego network of each node that sends the friend request, our approach would be able to point out the malicious behavior of some nodes. The new formed ECN around each ego node takes in to account nodes that participate in the community. The community does not include the friend request as an edge because it has not been accepted yet.

The implementation of algorithm 10 demonstrates that communities (groups) can be built by joining small sub-communities (subgroups) rather than considering the network as a whole. The detailed description of this method can be explained in two phases. On the first phase, for each user ($i \in V$) that sent a friend request to target(s), we aim to identify which users in the friend list ($v \in FL(u)$) of i would be tested first to form the ECN within a range 1 of i. This would be done by using a well-defined selection process as explained below. While on the second phase, after the selection process, we aim to check the feasibility of each node to be added to the ECN formed around i by utilizing an objective function (Section 6.5).

Algorithm 10 description: This algorithm takes as input the following two parameters: FR represents the set formed by all the users $i \in V$ that sent friend requests (FR) to the target t; and w is the weight between each edge. As a result, we would obtain the set ECN_{Total} which would contain all the ego centered networks of users who sent friend requests to the target t. We initialize ECN_{Total} as an as empty set *(lines 1)*. Between all the users that sent friend requests, we do not consider the ones that are not toward the target *(lines 2-5)*. Based on the ones in regard to the target, ECN_i which defines the ego centered network formed around the user i that sent friend request to the target is initialized to the user i itself, while S is initialized as the friend list of the user i that sent the friend request to the target *(lines 6)*. After, for each user that is in the friend list of i, we compute a similarity score (SBS) based of the user behavior and sparsity (equation 7) in respect to ECN_i and store it by decreasing priority/value in the list $Priority_z$ *(lines 7-10)*. This selection process, $Priority_z$, would allow us to first test the node with the highest

```
Algorithm 10: Ego-centered network detection
Input  : List of sent friend requests: $FR = \{(u,t) : u$ sends a friend
         request to $t\}$, Link weight: $w$.
Output: A set of ego-centered networks
         $ECN_{Total} = \{ECN_i : i = 1, ..., n\}$.
1  $ECN_{Total} \leftarrow \emptyset$
2  for $each\ node\ i \in V$ do
3      if $(i,t) \notin FR$ then
4          continue
5      end
6      $ECN_i \leftarrow \{i\}$, $S \leftarrow FL(i)$
7      while $|S| > 0$ do
8          for $z \in S$ do
9              $Priority_z \leftarrow SBS(ECN_i, z)$
10         end
11         for $z \in S$ in decreasing order in $Priority_z$ do
12             while $\mathcal{Q}(ECN_i \cup z) \leq \beta$ do
13                 $ECN_i \leftarrow ECN_i \cup \{z\}$
14                 $S \leftarrow S \setminus \{z\}$
15                 break
16             end
17         end
18     end
19     $ECN_{Total} = ECN_{Total} \cup \{ECN_i\}$
20 end
21 return $ECN_{Total}$
```

significance and further whether it would be added to ECN_i. Moreover, based on the node classification obtained by $Priority_z$, we use the objective function (Section 4.3) as another process to add a node from the selection procedure to ECN_i. Once a node z satisfies the condition, it is added to ECN_i to form a new ECN with i and also removed from the user i friend list S. Furthermore, the process starts over again from *(lines 6)* until all the nodes in S form user i have been tested. By adding one node at the time to ECN_i, this process allow us to form ECN not only around the user i and its neighbors but also around a growing ECN between user i and each node that is added in. We accomplish this process in respect to hyper-parameter β that helps us achieve the appropriate cohesiveness *(lines 11-18)*. We emphasize the ECN_i formation of a user i can be terminated once all the users in its friend list $FL(i)$ have been tested *(lines 7)* and do not satisfy the objective function *(lines 12)*.This process of ECN formation *(lines 2-18)* would be done on each user i that sent a friend request to target t to form the set ECN_{Total} *(lines 19)*. Finally, the algorithm returns ECN_{Total} which includes the ECN of each user i that sent a friend request to the target t *(lines 21)*.

Meanwhile, the propose similarity-based behavior structure (SBS) metric which determines which node satisfied the characteristic of cohesiveness can be defined as follows:

$$SBS(ECN_i, k) = \frac{|MF(ECN_i, k)|}{|FL(ECN_i)|} \times \frac{\sum_{u \in ECN_i, (u,k) \in E} w(u, k)}{\sum_{(u,v) \in FL(i), (u,v) \in E} w(u, v)} \tag{6.8}$$

where MF, FL, and ECN_i are described in Section 4.1 and k in a node inside $FL(i)$ ($k \in FL(i)$). $\sum_{u,v \in C} w(u, v)$ denotes the sum of friendship weights between nodes inside the community $FL(i)$ with $(u, v) \in E$.

6.6 Proposed Method for Detection of Colluding Groups in Targeted Reconnaissance Attack

In this section, we propose an overlapping community detection method along with some metrics to detect colluding friend request in TRA.

Colluding detection in Targeted Reconnaissance Attack

The purpose of the previous section was to identify the ego-centered network formed by users (i.e., $1, 2, ..., n$) that sent friend requests to the target t. The process was done by adding nodes that participate in the social circle of each user who sent friend request to the target. That allow us to point out the behavior associated to the interactions between colluders in targeted reconnaissance attack. While propagating in the network in the aim to befriend specific target(s), colluders involve in targeted reconnaissance attack do not participate in the social circle of the victims they befriended throughout the infiltration process. Therefore, this would make them vulnerable to our proposed approach by observing an anomaly behavior on colluders. Once the target is uncovered, colluders may befriend sets of common friends around the target to appear legitimate and even participate in each of their friend social circles. In addition, throughout infiltration attack, rather than befriend unknown users colluders focus on users that share common similarities (hobbies/interests, the closeness of mutual friends, school attended, etc.); it has been found that common similarity significantly influence users' decisions regarding friend requests [RBJB14]. By doing so, colluders involve in targeted reconnaissance attack build their own friend network, avoiding detection from Sybil defense mechanisms [BBR13, FGE14] or detection based on monitoring the networks using social rejection in OSNs [CSYM15]. Hence, our proposed detection mechanism should be strong enough to overcome such attack in the aim to prevent specific target(s) to accept malicious friend requests. The proposed algorithm and metrics to retrieve hidden colluders in targeted reconnaissance attack are described in detail as follow:

Algorithm 11 description: The input of algorithm 2 contains one main parameter describes as follows: ECN_{Total} from the output of algorithm 1 represents the main part in algorithm 2 to allow us to uncover the colluders involve targeted reconnaissance attack. At the end, we would acquire the set SC which consists of colluding users that sent friend requests to a specific target. We initialize ECN_{Total} as the set of all ego-centered networks formed by users ($i \in 1, ..., n$) that sent friend

Algorithm 11: Colluding Detection in TRA

```
Input  : A set of ego-centered networks : $ECN_{Total}$
Output: A set of colluders : $SC$
$ECN_{Total} \leftarrow \{ECN_1, ECN_2, ...ECN_n\}$
$SU \leftarrow \emptyset$
$M = \sum_{i=1}^{n} |ECN_i|$
for $i \leftarrow 1$ to $M - 1$ do
    for $j \leftarrow i + 1$ to $M$ do
        $SimScore \leftarrow TestSim(i, j)$
        if $SimScore \geq \gamma$ then
            $SU \leftarrow SU \cup (i, j)$
        end
    end
end
$N = \sum_{l=1}^{q} |SU_l|$
for $l \leftarrow 1$ to $N - 1$ do
    $x, y \in N_l$
    for $m \leftarrow l + 1$ to $N$ do
        if $x \vee y \in N_m$ then
            $OutScore \leftarrow |TestSim(x, z) - TestSim(y, z)|_{z \in N_m}$
            if $OutScore \leq \varepsilon$ and $FrienshipScore(i, z)_{i \in SC, z \in N_m} \geq \lambda$
             then
                $SC \leftarrow SC \cup \{x, y, z\}$
            end
        end
    end
end
return $SC$
```

requests to a target *(line 1)*. SU is initialized as empty set and represents the pairs of users $ECNs$ that have a higher similarity score $SimScore$ *(line 2)*. In addition, we store the number of ego-centered networks of users $(i = 1, ..., n)$ that send friend requests to specific target in M. This would allow us to control the comparison between pairs of users $ECNs$ *(line 3)*. After, we compute the similarity score $SimScore$ between all the possible pairs of $ECNs$ (i.e., between ECN_i and ECN_j). As a result, the pairs of users from which similarity scores that are greater than the hyper-parameter γ and that have not been tested yet would be store in SU *(line 4-11)*. Considering we take care of the order of comparison at *line 3*, we emphasize the duplication would not be observed during the computation of similarity scores between users $ECNs$. In order word, since $TestSim(i, j)$ is equal to $TestSim(j, i)$, the set (j, i) does not need to be added to SU. By doing so, SU would only include unique pairs of users $ECNs$. Before trying to identify hidden colluders by their network structure similarity, we store the number of ego-centered network pairs that have a higher similarity score $SimScore$ in N *(line 12)*. At this point, we need to identify hidden colluders by their network structure similarity. Based on the pair of users with the highest similarity score, we would add other users that satisfy predetermine conditions as colluders. For the pair of users with the highest similarity score in SU, we have users with $ECNx$ and $ECNy \in SU_l$ *(lines 13 - 14)*. Then for other pairs in CU, if one of the users x or y belongs to CU_m, we would compare $ECNx$ or $ECNy$ with the second user ECN in SU_m (e.g., $z \in SU_m$) by computing their $OutScore$ *(lines 15-17)*. Next, to unveil hidden colluding users involve targeted reconnaissance attack, we consider both $OutScore$ and $FriendshipScore$ metrics in respect to the hyper-parameters ε and λ and add users in the set of colluders (i.e., $SC \leftarrow \{x, y, z\}$). In this specific part (i.e., first round), all users (i.e., x, y, z) would be added but in the next and further round, only the second user of

SU_m would be added to SC. For instance, if the colluding group is $[a, b, c]$ and we found $(c, d) \in SU$, to test user d (i.e., determine if user is legitimate or colluder) behavior, we need to point out the interactions between d and the members inside the colluding group. Therefore, the following two conditions need to be satisfied as follow: first, $|TestSim(c, d) - TestSim(b, c)| \leq \varepsilon$ and $FrienshipScore(c, d) \geq \lambda$ *and* $FrienshipScore(b, c) \geq \lambda$; second, $|TestSim(c, d) - TestSim(a, c)| \leq \varepsilon$ *and* $FrienshipScore(c, d) \geq \lambda$ *and* $FrienshipScore(a, c) \geq \lambda$. The more users in SC the more tests need to be satisfied on the procedure. We emphasize on the adding process of each user in the set SC. For a new user to be considered and added as colluder in SC, the user would have to present the similar characteristics on $OutScore$ and $FriendshipScore$ with all the existing users already in SC *(lines 18-20)*. Finally, the algorithm returns a set of colluding group of users that sent friend requests to a specific target while trying to avoid detection by OSN mechanisms *(line 24)*.

Uncover Hidden Colluding Groups in Targeted Reconnaissance Attack

TestSim. This metric is based on the network topology (i.e., knowledge and share link between their friends) of the users who sent friend requests to the target. This metric considers two important characteristics between two users (e.g., u and v), the number of mutual friends and the number of different friends. The two assumptions we made here can be stated as follow: (i) the more mutual friends two users have, the more they know about each other; (ii) the more the links between different friends of two users, the more those two users may know each other. This metric aims to capture the connection behavior that colluding group in TRA will have; for instance, colluders may be connected to a set of nodes while avoiding to send friend requests to each other. As a result, this metric will best fit this behavior. Our proposed metric $TestSim$ can be defined as follow:

$$\begin{aligned} TestSim(u,v) = & \frac{|M(u,v)|}{|(FL(u) \cup FL(v))|} \\ & + \frac{|(a,b) \in E | \exists a,b : a \in DF(u,(u,v)) \wedge b \in DF(v,(u,v))|}{|DF(u,(u,v))| \times |DF(v,(u,v))|} \end{aligned} \tag{6.9}$$

$|M(u,v)|$ denotes the number of mutual friends between the users u and v, where $|(FL(u) \cup FL(v))|$ denotes the total number of friends between the users u and v. Also, $|DF(u,(u,v))|$ represents the number of different friends betwen the user u in respect to (u,v) and can be calculated as $|DF(u,(u,v))| = (|FL(u)| - |M(u,v)|)$; similarly, $|DF(v,(u,v))| = (|FL(v)| - |M(u,v)|)$. Finally, the numerator of the second characteristic represents the links between different friends of the users u and v.

FriendshipScore. This metric will consider users who have a high number of mutual friends in their $ECNs$ to reflect the similarity of both users. It can be defined as follow:

$$FrienshipScore(u,v) = (\frac{|M(u,v)|}{|FL(u)|} + \frac{|M(v,u)|}{|FL(v)|})/2 \tag{6.10}$$

Figure 6.2 gives an illustration and a better overview of the computation of the proposed metrics. We omit to represent the link weight of each edge for better clarity. The following estimation of our proposed metrics can help us to uncover a hidden colluding group of users who sent friend requests to a specific target.

$|FL(A_1)| = (1,2,3,4,8,a,b,d)$.

$|FL(A_2)| = (1,2,3,4,5,a,c,d)$.

$|M(A_1,A_2)| = (1,2,3,4,a,d)$.

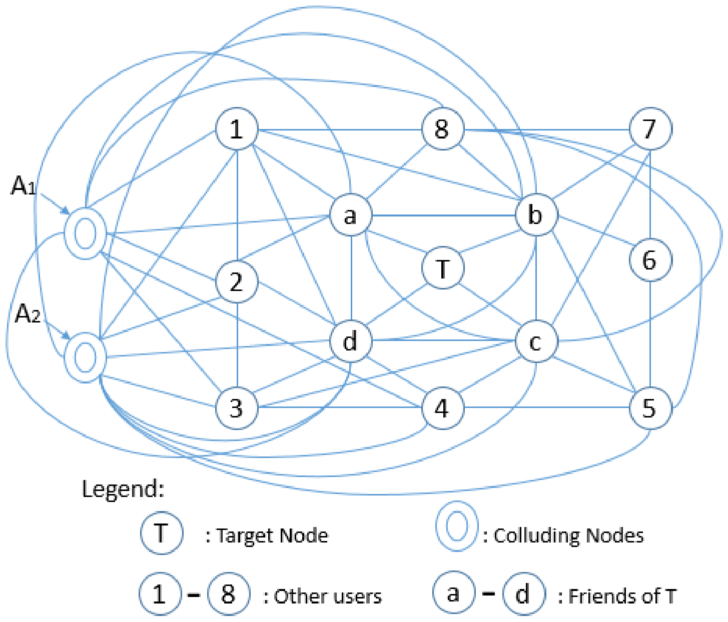


Figure 6.2: An illustration of $TestSim(A_1, A_2)$ and $FrienshipScore(A_1, A_2)$ in OSN.

$|TF(A_1, A_2)| = (1, 2, 3, 4, 5, 8, a, b, c, d)$.

$|DF(A_1, (A_1, A_2))| = (8, b)$.

$|DF(A_2, (A_1, A_2))| = (5, c)$.

$TestSim(A_1, A_2) = \frac{6}{10} + \frac{4}{4} = 1.6$

$FriendshipScore(A_1, A_2) = (\frac{6}{8} + \frac{6}{8})/2 = 0.75$

Therefore, we can conclude this section by emphasizing that our proposed metrics can be able to achieve a good result in detection of users involve in colluding targeted reconnaissance attack.

6.7 Simulation Results and Analysis

In this section, we first present the different network used in colluding targeted reconnaissance attack. After, we describe four standard indicators for the evaluation of our detection mechanism. Then, we evaluate the performance our proposed ego-centered network approach against other communities detection algorithms.

Experimental Setup

Our objective is to evaluate the proposed approach on ground-truth data. Therefore, we select three networks with different characteristics and give their description in Table 6.1.

Network	Type	Nodes	Edges
Facebook	Social Network	60k	1.55M
Enron Email	Communication Network	87k	322k
UC Irvine Email	Communication Network	1.9k	20k

Table 6.1: Datasets

All of the datasets [kon17b, VMCG09b, kon17a, KY04, kon17c, OP09] used in our simulation include direct networks (Table 6.1) and can be described as follows: a node and a directed edge represent a user and a sent message, respectively. For this type of network, the weight on each edge can represent the number of exchange messages between pair of users. Initially, all the networks are directed and it could be simple for malicious users involved in the collusion attack to exploit/manipulate their own directed edges. We follow the procedure in [WZG17, GWG$^+$18, BLS$^+$16] to change all directed networks to undirected ones by only considering undirected edge (u, v) between users u and v if both of them have existing directed edges (u, v) and (v, u), respectively.

As mentioned above, our proposed approach aims to detect malicious friend requests send to target by including users topology and interactions on OSNs. Therefore, the datasets need to contain weighted networks including both legitimate and malicious users. The weight between edges represents users interactions such as messages or emails. As a results of the inaccessibility of such datasets (i.e.,legitimate and malicious users), we generate a synthetic network for each dataset to represent groups of users by giving parameters such as numbers of nodes and average degree of nodes. This is done to satisfy a real-world scenario. We consider the nodes in the ground-truth datasets (Table 6.1) as legitimate and randomly choose a target user from the average connected users in each network. We also insert additional nodes to represent the malicious behavior. In order to create the behavior of malicious users involve in colluding targeted reconnaissance attack, we use the Preferential Attachment (PA) model [BA99] to generate a first network of 1000 nodes and randomly connect it to each legitimate network. In this network generated by PA, an edge is assigned a random weight corresponding to a value around the average weight of each legitimate network. We generate a second set of 30 isolated nodes that we randomly connect to nodes in each PA model network. Also, we add edges between neighbors of the target and the 30 nodes uniformly at random; this is to avoid rejection by target [LSDT16]. The edge weight value between the first and the second synthetic network or each legitimate network is lower than the one in each corresponding PA model network or legitimate network, respectively. Finally, a total of 1030 nodes is added to each legitimate network. We would randomly select 70 out of the 1000 nodes as legitimate friends requests while all the 30 nodes of the second synthetic network would represent malicious friend requests. Remove if existing edges between malicious nodes. Malicious users involved in colluding targeted reconnaissance attack do not built isolated group of friends and by removing friendship edges between colluding users, we want to relax the limitations of Sybil detection mechanisms that take the assumptions of existing edges between Sybil users and fewer attack edges (connections) between Sybil users and legitimate users in the network [GWG$^+$18, GGK$^+$15]. Therefore, we apply our detection approach to each ground-truth dataset combine with the two synthetic networks to identify malicious friend requests send to specific target.

Evaluation Metrics for our simulation

The purpose of this subsection is to present how well the proposed methods performed in detecting colluding groups in TRA on friend requests sent by multiple users to a specific target in OSNs. Some approaches consider the friend spam problem or Sybil detection as a classification problem that predicts whether a friend request sent by a user is malicious or not; and the Algorithms were evaluated in terms of True Positive Rate, False Positive Rate, and Accuracy [CSYM15, YXY$^+$16, WZG17]. True Positive Rate (TPR) can be defined in terms of true positives (TP) and false positives (FN) as mentioned in equation 11 [DG06]. False Positive Rate (FPR) is the result of false positives compared to true negative (equation 12) [DG06].

$$TPR = \frac{TP}{TP + FN} \tag{6.11}$$

$$FPR = \frac{FP}{FP + TN} \tag{6.12}$$

The definition for each indicators is as follows:

True positive (TP): result indicates that given colluding friend requests are indeed detected as colluder.

True negative (TN): result indicates that given legitimate friend requests are indeed detected as legitimate.

False positive (FP): result indicates that given legitimate friend requests are mistakenly detected as colluder.

False negative (FN): result indicates that given colluding friend requests are wrongly detected as legitimate.

The above-described indicators are commonly used to analyze the efficiency of an algorithm from various aspects. TPR estimates the fraction of friend requests that are correctly identify as colluders [DG06]. A higher TPR denotes a smaller number of colluding friend requests that have not been detected (false negative). FPR defines the fraction of legitimate friend requests identify as colluders [DG06]. Our method first investigates users $ECNs$ on each friend request received by a target. Second, from that set of users $ECNs$, we identify the colluding group of users that do not share friendship links between them (i.e., Colluding TRA behavior).

Experimental Results

In this chapter, we focus only on approaches that leverage users social graph structure or users interactions since we want to emphasize on the advantage of using both approaches for better detection mechanisms. Works such as [ML14, LM12, LWC14] which are neither users social graph structure or users interactions approaches since they leverage profile information (i.e., hometown, college, degree program, occupation, etc,...). Therefore, we are not including them in our evaluation. We evaluated our proposed methods against the colluding group in TRA to protect specific target(s) in OSNs, and we will consider three alternate detection methods as described below:

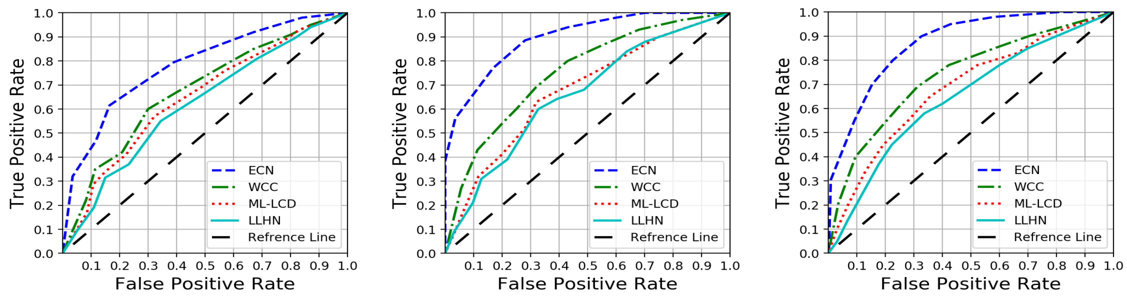
- WCC [OSKK05]: selects the seed node and measures the friendship formation between its neighbors. It builds community based on the contribution of each seed node friends in regards to all of its edge weights to form tightly connected neighborhood.
- ML-LCD [ITI$^+$16, ITI$^+$17a, ITI$^+$17b]: selects centered seed node to build a community around it based on a local approach that includes friends of the seed node within a range 1. It expands a community starting from a seed node

by performing an iterative search that seeks to maximize the ratio between the internal and external connections.

- LLHN [MBC17]: selects seed node friends and assigns a high similarity to node pairs that have many common neighbors compared not to the maximum number possible, but to the expected number of such neighbors. It adds the node with the highest score to form its community until the condition does not hold.

Our objective is to show that by using the proposed defense mechanism, the target can detect which friend requests are coming from groups that are either legitimate or malicious in OSNs. By utilizing the dataset [kon17b, VMCG09b, kon17a, KY04, kon17c, OP09], we modify the value of the threshold (i.e., SimScore) parameter into small variations to attain a deeper knowledge of how delicate it impacts the performance of the proposed methods. We compare our proposed ego-centered network approach against the others mentioned above. For illustration, we plot the Receiver Operating Characteristic (ROC) curve to show the performance of our proposed model at various thresholds values. ROC describes how accurate is a model on classification problem. A higher ROC indicates the model is better to classify friend requests between malicious and legitimate users.

Figure 6.3 represents the performance of different models on a classification problem on different datasets with various thresholds settings. It is shown that our model presents significantly better performance than the others. However, the performance varies depending on the network. First, we observe the performance depends on how dense is the network and how important is the users interactions (i.e., weight). Second, we note that ML-LCD and LLHN always present the lower ROC curves. Figure 6.3 (a) illustrates the least significant performance of all of the

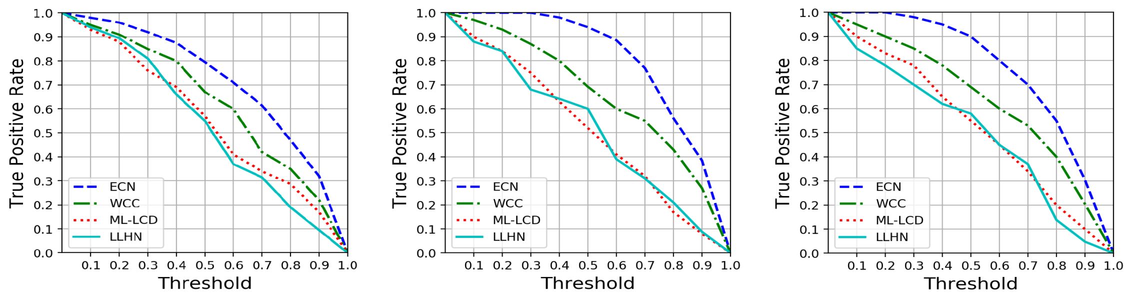


(a) UC Email (b) Enron Email (c) Facebook

Figure 6.3: Comparison between communities detection approaches on various networks.

models. This is caused by the sparsity and the lack of better users interaction in the dataset. Our model presents better results because of how it is able to accurately build the community around each users that send the friend request. ML-LCD and LLHN present bad results overall because they do not take advantage of the user interaction and only rely on the network topology. Even though WCC also presents better results than topological models (ML-LCD and LLHN), it still lacks to outperforms our model. This is caused by its ability to only consider users interactions to form community. Figure 6.3 (b) shows the importance of both topology and users interactions. On all of the models considered, ECN is strong enough to discourage potential colluders during target reconnaissance attack. By identifying more friend requests as colluders, our ECN model can achieve a better combination of precision-recall. Recall illustrates the percentage of all colluders in the friend requests while precision estimates the ratio of colluding friend requests that are indeed detected as colluders by a model. We also observe that ML-LCD and LLHN shows better performance than on Figure 6.3 (a) and are almost equal. We can infer that LLHN is able to capture the degree of each of user friend request common neighbors to form a better community in the aim to better detect more colluders. WCC presents a clear difference against topological models and its previous performance on Figure

6.3 (a). We can infer that WCC model is able to identify and capture the behavior associated to colluders. On Figure 6.3 (c), we once again observe how WCC does not perform well as in Figure 3 (b). We can infer the users interactions needed to build better community were not enough significant to help detecting colluders. We note that LLHN performs the worst and it is slightly outperforms by ML-LCD; we infer that ML-LCD better identifies nodes that maximize the ratio of connections to form better community to overcome malicious friend requests. Finally, ECN still gets better results overall. This can be explained by the method ECN model build each community for better colluding detection. Each community forms is not only centered on a user itself but also on each node that would be added to it. Therefore, this unveils the property of the triadic relationship observe in real-world while others (WCC and ML-LCD) rely on dyadic relationship which is not really practical.



(a) UC Email (b) Enron Email (c) Facebook

Figure 6.4: Colluding detection ratio on various networks.

Figure 6.4 illustrates the performance of different models against the similarity score (SimScore) on each built community around nodes that send the friend requests. It also addresses the difficulty of identifying malicious collaborative friend request (i.e., colluders) received by specific targets on various threshold settings. Figure 6.4 (a)-(c) confirm the results obtained in Figure 6.3 by giving a better illustration on the detection ratio of each model. Overall, it can be observed that

ECN outperforms others models. For instance, for a threshold of 0.7 on different datasets, we observe that ECN presents a true positive rate 0.615%, 0.77% and 0.7% for detection of malicious users involve in a colluding targeted reconnaissance attack, while other approaches were at 0.42%, 0.55% and 0.53% maximum respectively. We can conclude that users interactions can supplement a network topology for better detection. Even though one this two characteristics (topology or edge weight representing users interactions) is less significant in a network, the other can still helps with better detection. This has been observed in UC email, where users interactions is not as much important as in Enron and Facebook; but ECN still performs better in that condition.

6.8 Summary

In this chapter, we studied the problem of colluding targeted reconnaissance attack via integrating both graph structures and user behaviors in Online Social Networks [KZP+19]. This problem is challenging because colluders present different behavior that needs to be addressed, such as not sending friend requests between each other and creating a social circle around them to appear genuine to other users on the network [KPI+17a]. Therefore, we first proposed an approach based on a new objective function. We described some related algorithms to it to detect the ego-centered network with radius 1 of each node that sent the friend requests to the target. Specifically, we developed a Similarity-based Behavior Structure (SBS) that combines graph structures and users behaviors to approximate the closeness between node pairs. Based on the ego-centered network of nodes that sent friend requests to the target, we designed an approach to detect colluders in targeted reconnaissance attacks that relies on our newly developed metrics. We demonstrated that including both the graph structures and user behaviors results in a stronger detection mechanism to defend against colluders. Finally, the use of true positive rate, false positive rate and Recall for evaluation shows that our approach outperforms others [MBC17, ITI$^+$16, ITI$^+$17a, ITI$^+$17b, OSKK05]. The main reason, our defense mechanism achieves good outcomes, is that users behaviors are correlated with real-life scenarios, and our approach takes into consideration this correlation to propose a better defense mechanism. We address the limitations and future directions of this work in the next Chapter.

CHAPTER 7

LIMITATIONS, FUTURE WORK AND CONCLUSION

In this dissertation we investigated the security of online social networks and crowdsourcing platforms by proposing defense mechanisms against colluding users that are more difficult to detect than single attackers. In this chapter, we discuss the contributions and limitations, highlight the future work and conclusion.

7.1 Limitations

Colluding Users Detection in Crowdsourcing

We proposed, developed, and evaluated a defense mechanism that includes both a community detection algorithm and a semantic similarity algorithm to detect deceptive reviews. Our approach is able to detect a complex collaborative attack known as non-overlapping collusion. In this type of attack, colluders avoid creating the same clique sizes and participate over many products reviews with different colluding groups; the objective of the colluders in this attack is to collect as much money as possible without regard to the quality of the reviews they submit. Colluders can evade the defense mechanisms by actually trying to modify reviews based on semantics, rather than merely copying other reviews. We proposed a hybrid similarity (FuzzySim) measure through extracting parts-of-speech (POS) patterns from a text and using lemmatization to bring textual differences closer together. We then presented our first method to demonstrate that communities (groups) may be well-described by a model that joins small sub-communities (subgroups), thereby transforming a list of reviews into a network of collusion. A second method was offered to identify the overlapping groups by first finding local sub-communities and then identifying all the communities in the network, which was shown to detect up to 86% of non adversarial colluders with up to 93% accuracy, outperforming other approaches. Note that we want to maximize recall to find groups of malicious users who provide deceptive reviews while minimizing the number of legitimate users classified as colluders.

As limitations of this work, we did not investigate how well the proposed methods will perform in different environments or the case wherein colluders produce unique reviews based on other frameworks. Types of collaboration behavior between colluders need to be addressed, such as selfish behavior. It can be observed in the cases where some colluders may not want to cooperate on all malicious feedback.

Colluding Identity Clone Attacks Detection

We address the major question of whether friend requests sent to a specific target were coming from a colluding identity clone attack or legitimate users in online social networks. Colluders in identity clone attack (ICA) would attempt to recreate the social circles of the targeted user to send convincingly real friend requests; the purpose of this attack is to clone an existing user to infiltrate a particular social circle to gather information normally shared to the trusted users. We propose a threat model of colluding ICA that allows us to cope with real-world user's behavior in the case of collaborative malicious friend request sent to a specific target in the network, as well as a learning model that includes a three-step approach to detect colluding identity clone attack in online social networks. Our method exploits publicly available user features by way of a similarity algorithm that detects the similarities between different users. We highlighted the limitations of the original Monge-Elkan (e.g., asymmetric property and semantic relationships between words) and proposed a modified similarity algorithm, called FuzzySim, that exceeds those limitations and outperforms alternative algorithms that are based on vectorial similarity. Our classification method compares sets of profile pairs from different OSNs based on classification techniques that includes features extracted from each user's friend request information and their friend list. Based on the results of our comparison, we used a binary classification to rank the probability that the friend requests received by each user from the perpetrators of the colluding attack. The experimental results present that our proposed methodology can detect 80% of misleading friend requests coming from members of colluding ICA.

There are many limitations related to this scheme. The lack of real-world datasets to properly evaluate the effectiveness of it. We also need to address the knowledge on the network topology by colluders; what would happen if colluders have limited information on the target and their friends. This can reduce the legitimacy of the colluders friend requests to the target. In order to avoid detection, colluders can change their behavior such as not sending friend requests altogether at once or can avoid recreating social circles of their target. Those questions would enhance our proposed approach against advanced attacks.

Advanced Colluding Strategy in Targeted Reconnaissance Attack

We discussed modeling and exploring attackers' techniques used to infiltrate a network without being detected by defense mechanisms in place, which are the keys to identifying and developing countermeasures to the attack. Reconnaissance Attacks (RA) use OSNs to send friend requests to users in the network to explore and gain topological information. Another important variety of this type of attack is the Targeted Reconnaissance Attacks (TRA), in which attackers selectively befriend users in the network to ultimately befriend a target set of users while avoiding detection by OSNs defense mechanisms. A successful attack will meet a specific criteria: (1) it uncovers the topology of the network, (2) it minimizes/limits the number of friend requests, (3) it discovers viable paths to the target. We proposed a novel colluding attack strategy, called a Colluding Targeted Reconnaissance Attack (CTRA) in OSNs, which uncovers a specific target with incomplete information of the network topology within a maximum number of friend requests sent. Colluders in TRA tend to share topological knowledge of the network and use sociological theory behind friendship formation to influence their victims. Based on the concept of homophily, we develop an information benefit metric, called Weighted Similarity (WS), that is able to evaluate the closeness between pairs of users. We propose a two-fold colluding attack strategy that balances between exploration and exploitation to uncover a target. The experiment results on real world dataset present the effectiveness of our attack strategy compare to others. For 2000 attack trials, CTRA presents a 100% success rate within 60 friend requests.

We have so far only considered one dataset. However, datasets from different OSNs will challenge our strategy design to be efficient on either directed or undirected networks. Colluders behavior needs to be considered; some colluders may prioritize either exploration or exploitation and each time share their knowledge with others. Another improvement is to consider different ways for attackers to send friend requests during the infiltration; it can be a batch sequence where all of them send and share topological information at each step.

Colluding Targeted Reconnaissance Attack Detection

We addressed the fundamental question of whether friend requests received by a target were malicious or legitimate and if members of a colluding targeted reconnaissance attack (CTRA) in online social networks have meddled or not. We proposed an objective function to detect the ego-centered network (ECN) within range 1 of each node that sends friend request. We developed a Similarity-based Behavior Structure (SBS) to evaluate the closeness between ECN pairs. Based on some newly developed metrics and SBS, we designed an approach that is able to identify CTRA. This scheme has the advantage of including both graph-based (topology) and behavior-based (communication) characteristics of each user. Finally, the experimental results show that ECN outperforms others models. For instance, for a threshold of 0.7 on different datasets, we observe that ECN presents a true positive rate 0.615%, 0.77% and 0.7% for detection of malicious users involve in a colluding targeted reconnaissance attack, while other approaches were at 0.42%, 0.55% and 0.53% maximum respectively. Thus we inferred that both graph structures and user behaviors offer a better defense mechanism.

However, we did not account the detection for the n-radius of an ego-centered network ($n > 1$). We need to relax the assumption that the target has knowledge on the social interaction between each user that send a friend request and their friends; this may not exist in real life scenario. Another improvement is to consider the participation of the colluders in their friends social circle; this may allow colluders to appear as legitimate.

Few interesting directions for future work include the detection of n-range ego-centered network (n > 1) and investigate how well the proposed approach will perform. We plan to make our detection approach dynamic to cope with the change of the network over time. This would add new constraints such as graph structure and user behavior/interaction. We would examine how to integrate them to improve our model. Colluders may also present different behaviors among them in a manner that does not alert defense mechanisms.

7.2 Future Work

Regardless of the progress mentioned above, there are many promising directions to reinforce this research area in today's world dominated by the security concerns surrounding the Internet of Things (IoT) and Big Data. We plan to extend the scope of collusion attack detection scheme in various directions in the aim to develop more comprehensive frameworks.

Future work may study the proliferation of Mobile Devices (MD) that led to the advent of a new paradigm, called Mobile Crowd Sensing (MCS), wherein MDs form participatory sensing networks. MCS allows multiple Mobile Device Owner (MDO) to share data related to their local knowledge that is useful and rich with contextual information, but MCS is subject to privacy and security concerns from MDOs and the Sensing Service Consumers (SSC) alike: MDOs can maliciously collaborate with other MDOs to inject errors in the reported sensed information to distort interpretations of a specific event or collude with untrusted SSCs to reveal private information about target MDOs. A trust chain mechanism can be developed to enforce the normal behavior of MDOs while sharing data.

Another interesting direction to be explored is sockpuppets: many questions arise on the legitimacy of online discussion communities because of the anonymity provided by some discussion platforms (e.g., Yahoo, stack overflow, etc.) to users. Collaborative sockpuppets can engage malicious behavior, such as opinion manipulation or vandalize content, to deceive others. The goal would be to develop a defense mechanism that combines users activity features, community features, and post features.

Looking further, we intent to study two important cybersecurity topics related to social media: behavioral and predictive analytics. With the rise of information sharing in social media between users, users can present different behaviors depending on the platform. Such behaviors include sharing, liking, befriending and posting information. We will analyze the content of social media (e.g., Twitter) to uncover fake news produce by collaborative users for malicious intent. A group of users is said to have similar behavior if they behave altogether in the same manner. For instance, a group of users on Twitter that utilizes a hashtag to promote false information or movements could be considered malicious. Therefore, to thwart cyberattacks from malicious users, real-time detection is needed and becomes even more challenging due to dynamic environments of online platforms.

7.3 Conclusion

Online social networks and crowdsourcing platforms have become increasingly valuable resources. They provide services such as connecting people, sharing information, offering services for monetary gain, etc. These platforms have also become subject to major cyberattacks by collaborative malicious users due to the influence they have on organizations and people's decisions. In this dissertation, we discussed our proposed approaches to make OSNs and crowdsourcing platforms more secure against colluding attacks. Four core problems relating to colluding cyberattacks were studied in our dissertation. First, we propose methods to identify colluding malicious feedback in crowdsourcing. Based on part of speech and lemmatization, the scheme highlights the semantic between words. We have shown that our defense mechanisms can be used to improve the reliability of feedback that organizations rely on. Second, based on publicly available profile features in OSN, we propose a three-step method to identify clone attack launched by collaborative users to deceive legitimate OSNs users. Third, we also introduced a new attack (CTRA) to infiltrate a network without being detected to study and understand attackers procedure in the aim for better defense mechanisms. We design a two-fold strategy attack that includes exploration and exploitation to uncover a target in the network within a maximum number of friend requests. Finally, we develop a detection mechanism to protect the personal information of legitimate OSNs users against colluding targeted reconnaissance attack. In this mechanism, we perform a classification between legitimate or malicious friend requests. Hopefully, the outcome of this research will motivate better future defense mechanisms.

[ACE$^+$13] Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. Sok: The evolution of sybil defense via social networks. In *2013 ieee symposium on security and privacy*, pages 382–396. IEEE, 2013.

[Aft18] Steven Aftergood. Social media in security clearance investigations. https://fas.org/blogs/secrecy/2018/03/social-media-clearance/, March 2018.

[Ahm18] Irfan Ahmad. How much data is generated every minute? [infographic]. https://www.socialmediatoday.com/news/how-much-data-is-generated-every-minute-infographic-1/525692/, June 2018.

[AI15] Mohammad Allahbakhsh and Aleksandar Ignjatovic. An iterative method for calculating robust rating scores. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):340–350, 2015.

[AIB$^+$12] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Seyed-Mehdi-Reza Beheshti, Norman Foo, and Elisa Bertino. Detecting, representing and querying collusion in online rating systems. *arXiv preprint arXiv:1211.0963*, 2012.

[AIB$^+$13] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Elisa Bertino, Norman Foo, et al. Collusion detection in online rating systems. In *Asia-Pacific Web Conference*, pages 196–207. Springer, 2013.

[AS$^+$94] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[BBR13] Yazan Boshmaf, Konstantin Beznosov, and Matei Ripeanu. Graph-based sybil detection in social and information systems. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 466–473. ACM, 2013.

[BLM$^+$17] Francesco Buccafurri, Gianluca Lax, Denis Migdal, Serena Nicolazzo, Antonino Nocera, and Christophe Rosenberger. Contrasting false identities in social networks by trust chains and biometric reinforcement. In *Cyberworlds (CW), 2017 International Conference on*, pages 17–24. IEEE, 2017.

[BLS$^+$16] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Lería, Jose Lorenzo, Matei Ripeanu, Konstantin Beznosov, and Hassan Halawa. Íntegro: Leveraging victim prediction for robust fake account detection in large scale osns. *Computers & Security*, 61:142–168, 2016.

[BMBR11] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th annual computer security applications conference*, pages 93–102. ACM, 2011.

[BSBK09] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM, 2009.

[CGH18] Carol Cadwalladr and E Graham-Harrison. The cambridge analytica files. *The Guardian. Retrieved Mar*, 17:2018, 2018.

[Cis18] Cisco. Cisco 2018 annual cybersecurity report. https://www.cisco.com/c/en/us/products/security/security-reports.html, February 2018.

[CLSQ15] Kang Chen, Guoxin Liu, Haiying Shen, and Fang Qi. Sociallink: utilizing social network and transaction links for effective trust management in p2p file sharing systems. In *Peer-to-Peer Computing (P2P), 2015 IEEE International Conference on*, pages 1–10. IEEE, 2015.

[CNK$^+$17] Yizheng Chen, Yacin Nadji, Athanasios Kountouras, Fabian Monrose, Roberto Perdisci, Manos Antonakakis, and Nikolaos Vasiloglou. Practical attacks against graph-based clustering. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1125–1142. ACM, 2017.

[CRF03] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd* *workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[CSYM15] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Kamesh Munagala. Combating friend spam using social rejections. In *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*, pages 235–244. IEEE, 2015.

[CYYP14] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 477–488. ACM, 2014.

[DG06] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[DGMF15] Jörg Daubert, Tim Grube, Max Mühlhäuser, and Mathias Fischer. Internal attacks in anonymous publish-subscribe p2p overlays. In *Networked Systems (NetSys), 2015 International Conference and Workshops on*, pages 1–8. IEEE, 2015.

[DY11] Haitao Du and Shanchieh Jay Yang. Discovering collaborative cyber attack patterns using social network analysis. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 129–136. Springer, 2011.

[EDF$^+$] Sergey Edunov, Carlos Diuk, Ismail Onur Filiz, Smriti Bhagat, and Moira Burke. Three and a half degrees of separation.

[EFKE12] Aviad Elishar, Michael Fire, Dima Kagan, and Yuval Elovici. Organizational intrusion: Organization mining using socialbots. In *Social Informatics (SocialInformatics), 2012 International Conference on*, pages 7–12. IEEE, 2012.

[EFKE13] Aviad Elyashar, Michael Fire, Dima Kagan, and Yuval Elovici. Homing socialbots: intrusion on a specific organization's employee using socialbots. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 1358–1365. IEEE, 2013.

[ESKV17] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Towards detecting compromised accounts on social networks. *IEEE Transactions on Dependable and Secure Computing*, (1):1–1, 2017.

[EY17] Suhendry Effendy and Roland HC Yap. The strong link graph for enhancing sybil defenses. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 944–954. IEEE, 2017.

[Fac19] Facebook. Facebook reports fourth quarter and full year 2018 results. https://investor.fb.com/investor-news/press-release-details/2019/Facebook-Reports-Fourth-Quarter-and-Full-Year-2018-Results/default.aspx, January 2019.

[FGE14] Michael Fire, Roy Goldschmidt, and Yuval Elovici. Online social networks: threats and solutions. *IEEE Communications Surveys & Tutorials*, 16(4):2019–2036, 2014.

[FP16] Michael Fire and Rami Puzis. Organization mining using online social networks. *Networks and Spatial Economics*, 16(2):545–578, 2016.

[FVD$^+$16] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[GFM14] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE Transactions on Information Forensics and Security*, 9(6):976–987, 2014.

[GGK$^+$15] Peng Gao, Neil Zhenqiang Gong, Sanjeev Kulkarni, Kurt Thomas, and Prateek Mittal. Sybilframe: A defense-in-depth framework for structure-based sybil detection. *arXiv preprint arXiv:1503.02985*, 2015.

[GL16] Neil Zhenqiang Gong and Bin Liu. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *USENIX Security Symposium*, pages 979–995, 2016.

[GL18] Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM Transactions on Privacy and Security (TOPS)*, 21(1):3, 2018.

[GMJM$^+$16] Hector Garcia-Molina, Manas Joglekar, Adam Marcus, Aditya Parameswaran, and Vasilis Verroios. Challenges in data crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):901–911, 2016.

[GWG$^+$18] Peng Gao, Binghui Wang, Neil Zhenqiang Gong, Sanjeev R Kulkarni, Kurt Thomas, and Prateek Mittal. Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2018.

[HCSS14] Bing-Zhe He, Chien-Ming Chen, Yi-Ping Su, and Hung-Min Sun. A defence scheme against identity theft attack based on multiple social networks. *Expert Systems with Applications*, 41(5):2345–2352, 2014.

[ITI$^+$16] R Interdonato, A Tagarelli, D Ienco, A Sallaberry, and P Poncelet. Local community detection in multilayer networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 1382–1383. IEEE, 2016.

[ITI$^+$17a] Roberto Interdonato, Andrea Tagarelli, Dino Ienco, Arnaud Sallaberry, and Pascal Poncelet. Local community detection in multilayer networks. *Data Mining and Knowledge Discovery*, 31(5):1444–1479, 2017.

[ITI$^+$17b] Roberto Interdonato, Andrea Tagarelli, Dino Ienco, Arnaud Sallaberry, and Pascal Poncelet. Node-centric community detection in multilayer networks with layer-coverage diversification bias. In *Workshop on Complex Networks CompleNet*, pages 57–66. Springer, 2017.

[Jar89] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[Jar95] Matthew A Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995.

[JCB$^+$16] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Catching synchronized behaviors in large networks: A graph mining approach. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(4):35, 2016.

[JCW+13] Long Jin, Yang Chen, Tianyi Wang, Pan Hui, and Athanasios V Vasilakos. Understanding user behavior in online social networks: A survey. *IEEE Communications Magazine*, 51(9):144–150, 2013.

[JLTJ12] Lei Jin, Xuelian Long, Hassan Takabi, and James BD Joshi. Sybil attacks vs identity clone attacks in online social networks. *ISA 2012*, 2012.

[JTJ11] Lei Jin, Hassan Takabi, and James BD Joshi. Towards active detection of identity clone attacks on online social networks. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 27–38. ACM, 2011.

[JWG17] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Random walk based fake account detection in online social networks. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 273–284. IEEE, 2017.

[KCJ14] Ashiqur R KhudaBukhsh, Jaime G Carbonell, and Peter J Jansen. Detecting non-adversarial collusion in crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[KCLS17] Srijan Kumar, Justin Cheng, Jure Leskovec, and VS Subrahmanian. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 857–866. International World Wide Web Conferences Steering Committee, 2017.

[KML+19] Georges A Kamhoua, Navneedh Maudgalya, Benjamin Liang, Niki Pissinou, Hussein Zangoti, Laurent Njilla, Charles Kamhoua, Kianoosh G. Boroojeni, and SS Iyengar. Advanced colluding targeted reconnaissance attacks with incomplete information. In *In Preparation for ACM Digital Threats: Research and Practice (DTRAP), 2019*, pages 1–25. ACM, 2019.

[kon17a] Enron network dataset – KONECT, April 2017.

[kon17b] Facebook wall posts network dataset – KONECT, April 2017.

[kon17c] Uc irvine messages network dataset – KONECT, April 2017.

[KPI$^+$17a] Georges A Kamhoua, Niki Pissinou, SS Iyengar, Jonathan Beltran, Charles Kamhoua, Brandon L Hernandez, Laurent Njilla, and Alex Pissinou Makki. Preventing colluding identity clone attacks in online social networks. In *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on*, pages 187–192. IEEE, 2017.

[KPI$^+$17b] Georges A Kamhoua, Niki Pissinou, SS Iyengar, Jonathan Beltran, Jerry Miller, Charles A Kamhoua, and Laurent L Njilla. Approach to detect non-adversarial overlapping collusion in crowdsourcing. In *Performance Computing and Communications Conference (IPCCC), 2017 IEEE 36th International*, pages 1–8. IEEE, 2017.

[KPIM11] Georgios Kontaxis, Iasonas Polakis, Sotiris Ioannidis, and Evangelos P Markatos. Detecting social network profile cloning. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 295–300. IEEE, 2011.

[KSGM03] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

[KY04] Bryan Klimt and Yiming Yang. The Enron corpus: A new dataset for email classification research. In *Proc. European Conf. on Machine Learning*, pages 217–226, 2004.

[KZP$^+$19] Georges A Kamhoua, Hussein Zangoti, Niki Pissinou, Laurent Njilla, Charles Kamhoua, Kianoosh G. Boroojeni, Concepcion Sanchez Aleman, and SS Iyengar. Combating colluding reconnaissance attacks in online social networks. In *In Preparation for Journal of Communications, 2019.*, pages 1–18. Wiley, 2019.

[Lew12] Jason Lewis. How spies used facebook to steal nato chiefs details. https://www.telegraph.co.uk/technology/9136029/How-spies-used-Facebook-to-steal-Nato-chiefs-details.html, March 2012.

[LF09] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[LFS17] Zhepeng Lionel Li, Xiao Fang, and Olivia R Liu Sheng. A survey of link recommendation for social networks: Methods, theoretical foundations, and future research directions. *ACM Transactions on Management Information Systems (TMIS)*, 9(1):1, 2017.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[LL16] Yuhong Liu and Na Li. An advanced collusion attack against user friendship privacy in osns. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, volume 1, pages 465–470. IEEE, 2016.

[LL18] Yuhong Liu and Na Li. Retrieving hidden friends: A collusion privacy attack against online friend search engine. *IEEE Transactions on Information Forensics and Security*, 2018.

[LM12] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[LNJ$^+$10] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM, 2010.

[Lop11] Raul HC Lopes. Kolmogorov-smirnov test. In *International encyclopedia of statistical science*, pages 718–720. Springer, 2011.

[LSDT16] Xiang Li, J David Smith, Thang N Dinh, and My T Thai. Privacy issues in light of reconnaissance attacks with incomplete information. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*, pages 311–318. IEEE, 2016.

[LSS13] Ze Li, Haiying Shen, and Karan Sapra. Leveraging social networks to combat collusion in reputation systems for peer-to-peer networks. *IEEE Transactions on Computers*, 62(9):1745–1759, 2013.

[LST17] Xiang Li, J David Smith, and My T Thai. Adaptive reconnaissance attacks with near-optimal parallel batching. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 699–709. IEEE, 2017.

[LWC14] Rui Li, Chi Wang, and Kevin Chen-Chuan Chang. User profiling in an ego network: co-profiling attributes and relationships. In *Proceedings of the 23rd international conference on World wide web*, pages 819–830. ACM, 2014.

[Lyn10] Alec Lynch. Crowdsourcing is not new-the history of crowdsourcing (1714 to 2010). *Retrieved February*, 26:2013, 2010.

[LYS08] Yuhong Liu, Yafei Yang, and Yan Lindsay Sun. Detection of collusion behaviors in online reputation systems. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1368–1372. IEEE, 2008.

[LZY$^+$07] Qiao Lian, Zheng Zhang, Mao Yang, Ben Y Zhao, Yafei Dai, and Xiaoming Li. An empirical study of collusion behavior in the maze p2p file-sharing system. In *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*, pages 56–56. IEEE, 2007.

[MAC$^+$11] Nurul Nuha Abdul Molok, Atif Ahmad, Shanton Chang, et al. Information leakage through online social networking: Opening the doorway for advanced persistence threats. *Journal of the Australian Institute of Professional Intelligence Officers*, 19(2):38, 2011.

[MBC17] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)*, 49(4):69, 2017.

[MCS$^+$06] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[ME$^+$96] Alvaro E Monge, Charles Elkan, et al. The field matching problem: Algorithms and applications. In *KDD*, pages 267–270, 1996.

[ME97] Alvaro Monge and Charles Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. 1997.

[MGSPL17] Luis Munoz-Gonzalez, Daniele Sgandurra, Andrea Paudice, and Emil C Lupu. Efficient attack graph analysis through approximate inference. *ACM Transactions on Privacy and Security (TOPS)*, 20(3):10, 2017.

[ML14] Julian McAuley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):4, 2014.

[MLG12] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, pages 191–200. ACM, 2012.

[MS17] Ahmed Ould Mohamed Moctar and Idrissa Sarr. Ego-centered community detection in directed and weighted networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 1201–1208. ACM, 2017.

[MSLC01] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[MW09] Winter Mason and Duncan J Watts. Financial incentives and the performance of crowds. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 77–85. ACM, 2009.

[ND16] Hung T Nguyen and Thang N Dinh. Targeted cyber-attacks: Unveiling target reconnaissance strategy via social networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pages 288–293. IEEE, 2016.

[New03] Mark EJ Newman. Ego-centered networks and the ripple effect. *Social Networks*, 25(1):83–95, 2003.

[NWCH14] Jianwei Niu, Lei Wang, Yixin Chen, and Wenbo He. Detecting collusive cheating in online shopping systems through characteristics of social networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 311–316. IEEE, 2014.

[OCCH11] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

[OCH13] Myle Ott, Claire Cardie, and Jeffrey T Hancock. Negative deceptive opinion spam. In *Proceedings of the 2013 conference of the north amer-*

ican chapter of the association for computational linguistics: human language technologies, pages 497–501, 2013.

[OP09] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163, 2009.

[OSH$^+$07] J-P Onnela, Jari Saramäki, Jorkki Hyvönen, György Szabó, David Lazer, Kimmo Kaski, János Kertész, and A-L Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the national academy of sciences*, 104(18):7332–7336, 2007.

[OSKK05] Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. Intensity and coherence of motifs in weighted complex networks. *Physical Review E*, 71(6):065103, 2005.

[PGP13] Florian Probst, Laura Grosswiele, and Regina Pfleger. Who will lead and who will follow: Identifying influential users in online social networks. *Business & Information Systems Engineering*, 5(3):179–193, 2013.

[PPG05] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Security and Privacy, 2005 IEEE Symposium on*, pages 49–63. IEEE, 2005.

[PPS14] Abigail Paradise, Rami Puzis, and Asaf Shabtai. Anti-reconnaissance tools: Detecting targeted socialbots. *IEEE Internet Computing*, 18(5):11–19, 2014.

[PSP15] Abigail Paradise, Asaf Shabtai, and Rami Puzis. Hunting organization-targeted socialbots. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 537–540. IEEE, 2015.

[PSP$^+$17] Abigail Paradise, Asaf Shabtai, Rami Puzis, Aviad Elyashar, Yuval Elovici, Mehran Roshandel, and Christoph Peylo. Creation and management of social network honeypots for detecting targeted cyber attacks. *IEEE Transactions on Computational Social Systems*, 4(3):65–79, 2017.

[RBJB14] Hootan Rashtian, Yazan Boshmaf, Pooya Jaferian, and Konstantin Beznosov. To befriend or not? a model of friend request acceptance on facebook. In *Symposium on Usable Privacy and Security (SOUPS)*, pages 285–300, 2014.

[Ros15] Everett Rosenfeld. FBI investigating central command twitter hack. https://www.cnbc.com/2015/01/12/us-central-command-twitter-hacked.html, Jan 2015.

[RSL$^+$17] Shailendra Rathore, Pradip Kumar Sharma, Vincenzo Loia, Young-Sik Jeong, and Jong Hyuk Park. Social network security: Issues, challenges, threats, and solutions. *Information Sciences*, 421:43–69, 2017.

[SCL$^+$13] Zifei Shan, Haowen Cao, Jason Lv, Cong Yan, and Annie Liu. Enhancing and identifying cloning attacks in online social networks. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, page 59. ACM, 2013.

[SCM11] Tao Stein, Erdong Chen, and Karan Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, page 8. ACM, 2011.

[SE13] Aditya K Sood and Richard J Enbody. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy*, 11(1):54–61, 2013.

[SE15] Vlad Sandulescu and Martin Ester. Detecting singleton review spammers using semantic similarity. In *Proceedings of the 24th international conference on World Wide Web*, pages 971–976. ACM, 2015.

[SH15] Sucheta Soundarajan and John E Hopcroft. Use of local group information to identify communities in networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(3):21, 2015.

[SLSL16] Haiying Shen, Yuhua Lin, Karan Sapra, and Ze Li. Enhancing collusion resilience in reputation systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(8):2274–2287, 2016.

[SMJ$^+$15] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. Evilcohort: detecting communities of malicious accounts on online services. USENIX, 2015.

[SOJN08] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.

[SP15] David E. Sanger and Nicole Perlroth. Iranian hackers attack state dept. via social media accounts. https://www.nytimes.com/2015/11/25/world/middleeast/iran-hackers-cyberespionage-state-department-social-media.html, Nov 2015.

[SSM$^+$16] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, Daesung Moon, and Jong Hyuk Park. A comprehensive study on apt attacks and countermeasures for future networks and communications: challenges and solutions. *The Journal of Supercomputing*, pages 1–32, 2016.

[SSS06] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[TKFS13] Beth Trushkowsky, Tim Kraska, Michael J Franklin, and Purnamrita Sarkar. Crowdsourced enumeration queries. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 673–684. IEEE, 2013.

[TKMS03] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[TNLM15] Alberto Tarable, Alessandro Nordio, Emilio Leonardi, and Marco Ajmone Marsan. The importance of being earnest in crowdsourcing systems. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2821–2829. IEEE, 2015.

[VAD08] Luis Von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[VBC$^+$14] Bimal Viswanath, M Ahmad Bashir, Mark Crovella, Saikat Guha, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Towards detecting anomalous user behavior in online social networks. In

23rd {USENIX} Security Symposium ({USENIX} Security 14), pages 223–238, 2014.

[VLC$^+$17] Nguyen Vo, Kyumin Lee, Cheng Cao, Thanh Tran, and Hongkyu Choi. Revealing and detecting malicious retweeter groups. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 363–368. ACM, 2017.

[VMCG09a] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009.

[VMCG09b] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proc. Workshop on Online Social Networks*, pages 37–42, 2009.

[VPGM11] Bimal Viswanath, Ansley Post, Krishna P Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 41(4):363–374, 2011.

[WKFF12] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

[WLF11] Jiannan Wang, Guoliang Li, and Jianhua Fe. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 458–469. IEEE, 2011.

[WWZZ14] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *USENIX Security Symposium*, pages 239–254, 2014.

[WZG17] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Sybilscar: Sybil detection in online social networks via local rule based propagation. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[WZG18] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Sybilblind: Detecting fake users in online social networks without manual labels. In

International Symposium on Research in Attacks, Intrusions, and Defenses, pages 228–249. Springer, 2018.

[YSKY09] Yafei Yang, Yan Lindsay Sun, Steven Kay, and Qing Yang. Defending online reputation systems against collaborative unfair raters through signal modeling and trust. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1308–1315. ACM, 2009.

[YWW$^+$14] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):2, 2014.

[YXY$^+$15] Zhi Yang, Jilong Xue, Xiaoyong Yang, Xiao Wang, and Yafei Dai. Votetrust: Leveraging friend invitation graph to defend against social network sybils. *IEEE Transactions on Dependable and Secure Computing*, 13(4):488–501, 2015.

[YXY$^+$16] Zhi Yang, Jilong Xue, Xiaoyong Yang, Xiao Wang, and Yafei Dai. Votetrust: Leveraging friend invitation graph to defend against social network sybils. *IEEE Transactions on Dependable and Secure Computing*, 13(4):488–501, 2016.

[ZG16] Zhiyong Zhang and Brij B Gupta. Social media security and trustworthiness: overview and new direction. *Future Generation Computer Systems*, 2016.

[ZXL18] Xiaoying Zhang, Hong Xie, and John CS Lui. Sybil detection in social-activity networks: Modeling, algorithms and evaluations. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 44–54. IEEE, 2018.

VITA

GEORGES ARSENE KAMTO KAMHOUA

2014-Now	Ph.D., Computer Science Florida International University Miami, Florida, U.S.A.
2012	M.S., in Physics: Electronics University of Dschang Dschang, Cameroon
2007	B.Sc., in electrical engineering University of Dschang Bandjoun, Cameroon

PUBLICATIONS AND PRESENTATIONS

1. Kamhoua, Georges A., Niki Pissinou, S. S. Iyengar, Jonathan Beltran, Charles Kamhoua, Brandon L. Hernandez, Laurent Njilla, and Alex Pissinou Makki. *Preventing Colluding Identity Clone Attacks in Online Social Networks.* In Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on, pp. 187-192. IEEE, 2017.

2. Kamhoua, Georges A., Niki Pissinou, S. S. Iyengar, Jonathan Beltran, Jerry Miller, Charles A. Kamhoua, and Laurent L. Njilla. *Approach to detect non-adversarial overlapping collusion in crowdsourcing.* In Performance Computing and Communications Conference (IPCCC), 2017 IEEE 36th International, pp. 1-8. IEEE, 2017.

3. Concepcion Sanchez Aleman, Niki Pissinou, Sheila Alemany, and Kamhoua, Georges A. *A Dynamic Trust Weight Allocation Technique for Data Reconstruction in Mobile Wireless Sensor Networks.* In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 61-67. IEEE, 2018.

4. Concepcion Sanchez Aleman, Niki Pissinou, Sheila Alemany, and Kamhoua, Georges A. *Using Candlestick Charting and Dynamic Time Warping for Data*

Behavior Modeling and Trend Prediction for MWSN in IoT. In 2018 IEEE International Conference on Big Data (Big Data), pp. 2884-2889. IEEE, 2018.

5. Kamhoua, Georges A., Navneedh Maudgalya, Benjamin Liang, Niki Pissinou, Hussein Zangoti, Laurent L. Njilla, Charles Kamhoua, Kianoosh G. Boroojeni and S. S. Iyengar *Advanced Colluding Targeted Reconnaissance Attacks with Incomplete Information*. In Preparation for ACM Digital Threats: Research and Practice (DTRAP), 2019.

6. Kamhoua, Georges A., Hussein Zangoti, Niki Pissinou, Laurent Njilla, Charles Kamhoua, Kianoosh G. Boroojeni, Concepcion Sanchez Aleman and S. S. Iyengar. *Combating Colluding Reconnaissance Attacks In Online Social Networks*. In Preparation for Journal of Communications, 2019.