Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Cross-Device Tracking of Employees with Social Networks

Author:
Yuen Siong Choo

Supervisor:
Dr. Sergio Maffeis

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Degree in Computing Science / Secure Software Systems of Imperial College London

September 2017

Contents

1 Introduction **2**
- 1.1 Motivation . . . 2
- 1.2 Scope . . . 3
- 1.3 Contributions . . . 4
- 1.4 Plan of the Report . . . 6

2 Cross-Device Tracking **7**
- 2.1 Deterministic Logged-In CDT . . . 8
- 2.2 Deterministic Shared Credential CDT . . . 8
- 2.3 Probabilistic CDT . . . 9
  - 2.3.1 Using Machine Learning . . . 9
  - 2.3.2 Tracking Online Behaviour . . . 11
  - 2.3.3 Tracking Social Networks . . . 13
  - 2.3.4 Tracking Travel Patterns . . . 15
- 2.4 Alternative CDT Techniques . . . 17

3 Technical Background **18**
- 3.1 Directed Triadic Closure . . . 18
- 3.2 Maximum Likelihood Estimation . . . 25
  - 3.2.1 Formula's Basic Concept . . . 25
  - 3.2.2 Formulating the Likelihood Function . . . 25
  - 3.2.3 Contextualising the Likelihood Function . . . 28
  - 3.2.4 Maximising the Likelihood Function . . . 29
- 3.3 Twitter . . . 31
  - 3.3.1 Twitter Proprietary Link Shortener . . . 31
  - 3.3.2 Twitter APIs . . . 32
  - 3.3.3 Limitations . . . 33

4 Threat Models **35**
- 4.1 Network Eavesdroppers . . . 35
- 4.2 Malicious Third-party Sites . . . 37

5 Implementation **39**
- 5.1 Deducing Employees' Twitter Accounts . . . 39
  - 5.1.1 Deducing the Most Likely User . . . 42
  - 5.1.2 Deducing Other Likely Users . . . 43

5.1.3 Ranking Deduced Employees 45
5.2 Verifying Anonymous PMDs 46

6 Evaluation **49**
6.1 Social Networks Dataset 49
6.2 Constructing Synthetic URLs 51
6.2.1 Company's URLs 51
6.2.2 Anonymous PMDs' URLs 52
6.3 Deducing Employees' Accounts 54
6.3.1 Top Likelihood Deduction 54
6.3.2 Eliminate & Iterate Likelihood Deduction 55
6.3.3 Ranking Deductions 55
6.4 Verifying Anonymous PMDs 56
6.5 Adversary Limitations 59

7 Countermeasures **62**
7.1 Switching Accounts to Private 62
7.2 Inducing Noise URLs 63
7.3 Decoy Countermeasures 64
7.3.1 Decoys Mirroring Employees' Followerships 64
7.3.2 Decoys Following Radom Users 66
7.3.3 Avoiding Decoys Detection 67
7.4 Alternative Countermeasures 67

8 Conclusion and Recommendations **69**

Appendices **78**
A Experiments Results 79

List of Figures

3.1 How *directed triadic closure* influence the building of relationships. . . 19
3.2 Scenario 1: Relationships between A,B,C,D,E and F before and after the recommendations. 20
3.3 Scenario 2: Relationships between A,B,C,D,E and F before and after the recommendations. 20
3.4 Scenario 3: Relationships between A,B,C,D,E and F before and after the recommendations. 21
3.5 A user's followees can be grouped according to a user's social circles. 22
3.6 An example of how a set of URLs observed from a company could be contributed by its employees. 22
3.7 Two possible end cases of how a set of URLs observed from a company could be contributed by its employees. 23
3.8 Social network structure of a company before and after *snowballing directed triadic closure*, and how a set of URLs observed from it could be contributed by its employees. 24
3.9 Illustration of the objective of the MLE formula discussed in (Su et al., 2017). 26
3.10 natural logarithm is a increasing function 27

4.1 Monitoring capabilities of a powerful network eavesdropper on fixed-line broadband. 36
4.2 Monitoring capabilities of a powerful network eavesdropper on mobile broadband. 36
4.3 Monitoring capabilities of a powerful third-party site. 38

5.1 Extracting *t.coURLs* from URLs observed from a company. 40
5.2 The user who have generated the observed URLs is likely to be a follower of the users who have posted Tweets containing the corresponding links. 40
5.3 Making consecutive requests to Twitter to find users that potentially could be one of the employees. 41
5.4 Retrieving the feeds for each of the follower that is derived from the observed *t.coURLs*. 42
5.5 Finding the likelihood that a subset of the *t.coURLs* in the feeds of a derived follower can be found in the set of *t.coURLs* extracted from the observed URLs. 42

5.6 Different possible scenarios when removing the observed *t.coURLs*. . . 44
5.7 Extracting *t.coURLs* from URLs observed from an anonymous PMD. . . 47

6.1 Examples of social network structures in Twitter social circles dataset from Stanford Large Network Dataset Collection. 50
6.2 Plot of *Top Likelihood Deduction* algorithm's TPRs vs clustering coefficients 54
6.3 Performance of *Eliminate & Iterate Likelihood Deduction* algorithm . . 55
6.4 Plot of deduction TPR vs top % of ranking. 56
6.5 Plot of TPRs vs number of anonymous URLs, for verifying anonymous PMDs belonging to employees. 57
6.6 Plot of TNRs vs number of anonymous URLs, for verifying anonymous PMDs that do not belong to employees. 58
6.7 Plot of % URLs and TPRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs that belonging to employees. 60
6.8 Plot of % URLs and TNRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs belonging to employees' followees. 61
6.9 Plot of % URLs and TNRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs belonging to employees' followers. 61
6.10 Plot of % URLs and TNRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs belonging to strangers. . 61

7.1 Plot of TPRs vs % of private Twitter Accounts, for each social circle . . 63
7.2 Plot of TPRs vs % of noise URLs, for each social circle 64
7.3 Redistributing employees' followees among a smaller group of decoys. 65
7.4 Plot of TPRs vs number of decoys, for each social circle. 66
7.5 Creating decoys that follow other random users who are not employees' followees. 67

List of Tables

3.1 Twitter REST API methods used in our implementations. 32

6.1 Size and clustering coefficients of 5 Twitter social circles randomly chosen from the dataset. 50
6.2 Breakdown of the number of users selected as active employees, and the total number of URLs constructed as URLs observed from the 5 social circles 52
6.3 Breakdown of the number of selected users in each of the 4 sets of anonymous users and the total number of URLs constructed as URLs observed from their PMDs 53

A.1 Performance of the *Top Likelihood Deduction* algorithm with different numbers values of N, for each social circle. 79
A.2 Performance of the *Eliminate & Iterate Deduction* algorithm with different numbers values of N, for each social circle. 80
A.3 Performance of ranking the deduced Twitter accounts with different numbers values of N, for each social circle. 80
A.4 Performance of the algorithm to verify anonymous PMDs for each social circle. Tests are conducted with $M = 50$ maximum randomly selected users from each set of anonymous user type. 81
A.5 Performance of the algorithm to verify anonymous PMDs when faced with the limitations of network eavesdroppers, for social circle no. 5. Tests are conducted with $M = 50$ randomly selected users from each set of user type. 82
A.6 Performance of the algorithm to verify anonymous PMDs when faced with the limitations of malicious third-party sites, for social circle no. 5. Tests are conducted with $M = 50$ randomly selected users from each set of user type. 83
A.7 Performance of the *Top Likelihood Deduction* algorithm with different percentage of private Twitter accounts, for each social circle. 84
A.8 Performance of the *Eliminate & Iterate Deduction* algorithm with different percentage of private Twitter accounts, for each social circle. . 84
A.9 Performance of the *Top Likelihood Deduction* algorithm with different percentage of noise URLs, for each social circle. 84
A.10 Performance of the *Eliminate & Iterate Deduction* algorithm with different percentage of noise URLs, for each social circle. 85

A.11 Performance of the *Top Likelihood Deduction* algorithm with different numbers of decoy Twitter accounts for each social circle, where $l \equiv |\text{social circle}|$. 85
A.12 Performance of the *Eliminate & Iterate Deduction* algorithm with different numbers of decoy Twitter accounts for each social circle, where $l \equiv |\text{social circle}|$. 85

Abstract

Companies around the world spent millions of dollars purchasing state-of-the-art network intrusion detection system to protect their network infrastructure from cyberattacks. Yet, they often neglect to protect employees' personal mobile devices (PMDs). Adversaries could hack employees' PMDs when they are outside the protection of the company's firewalls and potentially exploit them as trojan horses for indirect attacks: such as slipping in malware while employees charge their PMDs using office devices (ODs), or exploiting PMDs' bluetooth or NFC connection to infiltrate the company's network via vulnerable intermediaries, or using PMDs' cameras and microphones as eyes and ears in the company. Given the potential cybersecurity risks that compromised PMDs pose, it will then be important to protect them from being hacked by adversaries and the most straightforward method is to prevent adversaries from even identifying the employees' PMDs.

Su et al. (2017) suggest the possibility of using social media feeds to de-anonymise web browsing traffic, and in this report, we show that this concept can also be exploited to cross track employees' ODs and their PMDs. Similar to (Su et al., 2017), our approach is based on the idea that each person has a distinctive social network, and thus the links appearing in one's social media feeds are also unique. But on top of that, we further study how such behaviour evolves when it happens to a group of people belonging to the same social circle. We found that over time, due to the process of *snowballing directed triadic closure*, members of a social circle are likely to share similar followees and feeds which are unique to that social circle. This allows us to exploit the formula in Su et al. (2017) to 1) deduce employees' social media accounts using web browsing traffic observed from the company, and 2) verify if an anonymous PMD belongs to an employee using web browsing traffic observed from that PMD.

We implement and evaluate our strategy on synthetic web browsing traffic constructed from social circles that are derived from the Twitter social circles dataset from the Stanford Large Network Dataset Collection, and show that accurate deductions (i.e. TPR $>$ 0.8) and verifications (i.e. TPR and TNR $>$ 0.8) are achievable using our strategy. We also explore how the limitations faced by adversaries in real-life applications affect the performance of our strategy. Finally, we demonstrate how to counter our strategy using four different measures and assess their pros and cons.

Acknowledgements

I would like to acknowledge the help of all those who made this research possible. In particular, I would like to thank my supervisor Dr. Sergio Maffeis for his time, patience and guidance. I would also like to thank my examiner Dr. Peter Pietzuch for helping to assess my work. Finally, I would like to thank my wife, April for her unwavering support and encouragement throughout this period.

Chapter 1

Introduction

1.1 Motivation

Companies around the world spent millions of dollars purchasing state-of-the-art network intrusion detection system to protect their network infrastructure from cyberattacks. Yet, they often neglect to protect employees' personal mobile devices (PMDs) such as smartphones, tablets or even laptops. Employees' PMDs, outside the protection of the company's firewalls, are connected to vulnerable networks like mobile broadband, home LAN or public hotspots and are subjected to many types of cyberthreats that could potentially exploit them as trojan horses for indirect attacks. Some might argue that as long as such devices are not allow to connect to the company's network, it should not be a major concern. However, as NSA has shown in their efforts to hunt and hack system administrators (Gallagher and Maass, 2014), human are always the weakest links to a well defended fortress. For example, malware can slip through undetected while employees charge their PMDs using office devices (ODs); Adversaries can also exploit PMD's bluetooth or NFC connection to infiltrate the network via vulnerable intermediaries instead of connecting directly to the company's wifi; Attached cameras and microphones could also serve as adversaries' eyes and ears in the company.

Given the potential cybersecurity risks that compromised PMDs pose, it might be justifiable to impose strict restrictions on them within the company premises. However, such drastic measures will bring about significant limitations on the employees that could affect productivity, performance and even retention (Atkinson, 2016). A more feasible preventive measure would be to block the attacks at the initial stages, that is, prevent adversaries from even finding and hacking employees' PMDs. More specifically, make it either significantly cost-ineffective for adversaries to find employees PMDs or is only achievable with very low accuracies and confidences.

1.2 Scope

One might ask how an adversary is able to uniquely identify employees among the billions of PMDs users out there (Lunden, 2015). The "James Bond" method would be to physically track the employees, divert their attention and implant the malware onto their PMDs while they are distracted. However, in this modern Internet connected world, it is no longer necessary to embark on such risky operations just to hack a device: Adversaries such as network eavesdroppers and malicious third-party sites possesses the ability to remotely track cyber activities of internet-connected devices, and there exist many methods to remotely implant malware on them as well (Patange, 2013; Gallagher and Greenwald, 2014; Giandomenico, 2017). For example, adversaries could monitor DNS requests or VPN connections to the company's mail or VPN servers and exploit the initiating client using man-in-the-middle (Patange, 2013), man-on-the-side (Gallagher and Greenwald, 2014) or spear phishing attacks (Giandomenico, 2017) etc., just to name a few. They could also monitor for credentials or cookies that are shared between employees' ODs and their PMDs (Brookman et al., 2017).

One interesting concept that could potentially be used to track employees' PMDs is the uniqueness of one's feeds in social media services such as Facebook, Twitter and Weibo etc. Su et al. (2017) hypothesise that each person has a distinctive social network, and thus the links appearing in one's social media feeds are also unique. The authors further demonstrated that adversaries such as network eavesdroppers and malicious third-party sites can identify the urls of such links when the users visit (i.e click) them, and accurately deduce the users' accounts from them. Throughout the remainder of this report, we use the term URL to refer to the url of a link that is found in one's social media feeds and is visited by the user.

However, their strategy is more suitable for de-anonymising small number of URLs as they need to make several requests to the social media service for each of them. This will not be cost-effective for big-time adversaries who are potentially observing millions of URLs daily. Nevertheless, if the adversaries already know who they are targeting, it is possible to make these requests beforehand and restrict them only to those that are related to their targets. Adversaries could crawl social media services en masse to find potential employees' accounts, but as not all users would want to reveal their employment information or keep them updated regularly, a complementary strategy is required. For example, given that companies usually lease static IP addresses from their Internet service providers (ISPs), it is possible for adversaries to monitor URLs from a company-of-interest over an extended period of time to deduce the employees' accounts. However, any such strategy needs to overcome the challenge of differentiating URLs coming from many different employees, as they are all mixed together when they passes through the company network address translation (NAT) gateway (Wikipedia, 2017d). As such, the strategy used in (Su et al., 2017) will not be suitable for this purpose, and an alternate solution is required.

In this report, we show that it is still possible to use one's social media feeds to

accurately deduce employees' accounts, even when the observed URLs are a mixture coming from many different employees. This is possible because social media services tends to recommend users, other users who are currently being followed by their followees. This is known as *directed triadic closure* (Romero and Kleinberg, 2010) and over time as the effect snowball, it increases the chance that employees in a company will share similar followees. Therefore, even when the observed URLs are a mixture coming from many different employees, it is still a composition of URLs that are either posted by followees who are unique to individual employees or by those that are shared among the employees. If most of the URLs falls under the first scenario, there is little overlap and it will be fairly easy and accurate to deduce the employees involved, since each portion of the URLs (when sufficiently large) uniquely represent a user (Su et al., 2017). If most of the URLs falls under the second scenario where there are significant overlaps, we are still able to deduce at least one of the employees accurately because the overlapped URLs are posted by a set of shared followees that are unique to the company.

We also show that once the employees' accounts are known, it will be relatively cost-effective to accurately track them out in the wild using URLs observed from their PMDs. Cost-effectiveness is achieved by regularly making requests to social media services for updated feeds prior to matching them with the PMDs' URLs. By making the requests beforehand and caching the feeds, we reduce the time needed to verify each anonymous URL and thus a larger number of them can be verified in a much shorter time. Furthermore, since we only send requests that are related to a fixed number of employees, they can be done with relatively manageable and predictable amount of processing resources.

1.3 Contributions

Our report make four contributions:

- Firstly, we show that employees of a company tend to follow similar social media accounts because of *directed triadic closure*. *Directed triadic closure* influences social media services to recommend accounts that are being followed by the followees of a user, and the more followees that follow a recommended account, the more likely the user will follow that account. This effect will be more prominent among members of offline social circles such as colleagues, since they might know each other in person and hence has more trust in each other's choice of followees. Over time, employees will develop a set of followees that are uniquely shared by them and the URLs that these followees posted could be used to deduce the employees' accounts. We design an algorithm that implements the Maximum Likelihood Estimate (MLE) formula from (Su et al., 2017) to accurately deduce at least one of the employees even when

the observed URLs are random mixtures visited by many different employees.

- Secondly, we design a cost-effective algorithm to verify if a set of anonymous URLs are generated by a set of known employees. Our algorithm send request to social media services to cache updated feeds of each known employee prior to verifying any of the observed anonymous URL. By avoiding the need to send request at the point of verification, we are able to process much more URLs at any one time. Verification of anonymous URLs is achieved by including Likelihood Ratio Testings (LRT) on top of MLE calculations. This is because MLE score alone only allow us to rank and select the most likely candidate from the list of known employees, but it does not measure how accurate this selection is. For example, despite having only one of the anonymous URLs appearing in the feeds of the most likely candidate, it will still be selected when none of the other URLs appear in the feeds of other employees. Here, the most likely candidate is definitely not an accurate selection since majority of the anonymous URLs did not appear in his or her feeds. By incorporating an additional LRT to test MLE score against corresponding Maximum Unlikelihood Estimate (MUE) score, we can further determine if our selection is accurate.

- Thirdly, we design four preventive measures that can counter our strategy, thereby preventing adversaries from exploiting employees' PMDs to launch indirect attacks on the company and its networks. First, we show that by advising employees to keep their social media accounts private, adversaries will not be able to obtain their feeds and therefore, unable to use them to deduce their accounts. Second, we show that by inducing more noise into the observed URLs, the performance of our strategy will be affected. Our third and fourth countermeasures create decoys that either mirror employees' followerships or follow any other random users that are not related to the employees. Using decoys makes our countermeasures more socially responsible as it will lead adversaries to the decoys instead of other innocent users.

- Finally, we implement and evaluate our strategy using a popular social media service (Twitter), even though they can be used with any other social media services (as long as the employees' accounts are public and sufficiently many links from their feeds are visited on both their ODs and PMDs). We have chosen Twitter because, like what Su et al. (2017) have pointed out, their links are wrapped in a proprietary shortener (i.e. https://t.co) that could be easily identified by both network eavesdroppers and malicious third-party sites. Furthermore, most Twitter activities are public and it has a open-source APIs that we can use to retrieve essential information such as postings or followee/follower relationships etc. (Twitter, 2017a). However, the APIs has its limitations: it is rate-limited, third-parties are not able to requests for a user's feeds without the owner's authorisation, and its Search function only searches against a sampling of recent Tweets published in the past 7 days (Twitter, 2017a).

These limitations have restricted us in conducting comprehensive experiments involving more test subjects, but we do not foresee them restricting the adversaries whom we expect to be more resourceful than us and are more targeted in their usage.

1.4 Plan of the Report

The rest of this report is structured as follows. In Chapter 2 we review the related works. In Chapters 3 and 4, we discuss in more details the technical background of our strategy, and the various threat models respectively. Chapter 5 details our implementations, and we evaluate their performances in Chapter 6. We review the possible countermeasures in Chapter 7 and finally, Chapter 8 summaries our conclusions and recommends future work directions.

Chapter 2

Cross-Device Tracking

Our approach of finding employees is akin to cross-device tracking where cyber activities of a device is being monitored to obtain a set of credentials, profiles or even cyber behaviour models (e.g. browsing behaviours) that are representative of the device user. The underlying assumption is that these credentials, profiles and models would still be representative even when the user switches devices. Researches related to cross-device tracking techniques is an emerging field of study that has gain traction in the past few years. In this section, we briefly introduce key concepts and existing researches that are relevant to cross-device tracking.

Cross-device tracking refers to the act of tracking individuals across multiple devices, such as personal computers, tablets and smartphones etc. The objective is to build a comprehensive and accurate profiles of individuals by correlating information collected from the various devices that they used. Unlike tracking of individual devices, cross-device tracking aims to identify the person rather than a pseudonymous device and, as such, is potentially much more privacy-invasive than the tracking of unconnected devices.

However, privacy concerns arising from such acts only gain recognition in recent years. For example, the Federal Trade Commission (FTC) conducted a joint workshop with Office of Technology Research and Investigation (OTRI) to explore privacy issues concerning cross-device tracking in 2015 (Federal Trade Commission, 2015) and have only issued warning letters to app developers regarding the use of cross-device tracking without user consensus in 2016 (Federal Trade Commission, 2016). Digital Advertising Alliance (DAA) have also released formal guidance that only goes into effect in 2017 (Tonsager, 2016). Although such efforts are commendable, we should not take it for granted that they would eliminate the dangers associated with cross-device tracking as they will not prevent adversaries that operates outside the jurisdictions of FTC and control of DAA in conducting such activities.

Cross-device tracking is generally more challenging than same-device tracking, especially for network eavesdroppers and third-party sites. First party sites usually have the ability to track their account holders' activities regardless of device, using their login credentials. However, those that do not have such direct contractual re-

lationships will need to go around the limitation via alternative means. In general, cross-device tracking techniques can be broadly categorised into either 1) Deterministic Logged-In, 2) Deterministic Shared Credential and 3) Probabilistic (Brookman et al., 2017).

2.1 Deterministic Logged-In Cross-Device Tracking

In deterministic logged-in cross-device tracking, first-parties that allows their users to login to their personal accounts from different devices place cookies or device-specific identifiers to remember and track them across devices. This is usually advertised as a value-added security feature where they help users to keep track of logins attempted from a new device which might suggest a possible fraudulent activity. When such event happened, first-parties can either notify their users or require a secondary authentication before granting access to the account. Many of them also show their users the list of devices that have logged into their account, so that they can monitor potentially unauthorised accesses themselves.

Many first-parties also embed themselves in other operators' websites as social sharing widgets, analytics code, social login, or advertising etc. Therefore, if an already logged-in user visits such websites, the first-parties will be notified regardless of whether the user interacts with the embedded content. And if the user do this on multiple devices, the first-parties will be able to track him or her across those devices.

Deterministic logged-in cross-device tracking will not be employable by adversaries such as network eavesdroppers and malicious third-parties as they usually do not have direct contractual relationships with users.

2.2 Deterministic Shared Credential Cross-Device Tracking

In shared credential cross-device tracking, first-parties may share (Personally Identifiable Information) PIIs with third-parties, and if the same PII is shared from multiple devices, third-parties will also be able to track the logged-in users across those devices that they used. Many first-parties will cryptographically hash the PIIs that they shared with third-parties so that they do not violate the agreement with their users. However, this does not prevent third-parties from conducting effective cross-device tracking since the same PII will be hashed to same unique pattern (unless a hash collision occurs). This means that even when third-parties do not have a direct login relationship with users, they are still able to engage in cross-device tracking through contractual relationships with the first-parties.

Shared credential cross-device tracking could be employed by adversaries such as network eavesdroppers and malicious third-parties. However, as previously detailed in (Englehardt et al., 2015), such credentials are not substantial and its usage is opportunistic as it depends on websites transmitting such credentials. Therefore, this technique is usually not used as the primary attack mechanism, but rather as a complement.

2.3 Probabilistic Cross-Device Tracking

Probabilistic cross-device tracking works by correlating pseudonymous identifiers or behaviours collected from multiple devices to infer the likelihood of those devices belonging to the same person. These devices would usually share similar attributes like IP address or geolocation. For example, when two devices consistently have the same IP address across multiple time spans, they might be sharing the same WIFI network frequently and this increases the possibility that they are owned by the same person. However, like all probabilistic approaches, there will be situations that do not fit the hypothesis, such as when the devices are connected to public WIFI hotspots. In this case, not all connected devices will belong to the same user. Nevertheless, a recent study estimated that the accuracy of probabilistic cross-device tracking can reach as high as 97.3% even without PIIs (Bilton, 2015).

2.3.1 Using Machine Learning

Machine Learning plays a central role in probabilistic cross-device tracking and many prior works use it to study privacy issues arising from cross-device tracking. For example, Zimmeck (2017) explores various distance measures, such as Jaccard index, Cosine similarity and Bhattacharyya coefficient on features such as IP address, web domain and app-to-web domain, to determine the similarity between devices. In particular, he attempts to match mobile devices to desktops that are potentially used by the same user. His results indicate that similarities in IP addresses gives the highest probability that a mobile device and desktop belongs to the same user, which confirms the intuition that devices with similar IP footprints across time are more likely to be used by the same individual. Similarities in web domains are also good indicators, especially when using Bhattacharyya coefficient and excluding popular domains. This will be useful in situations where several users share the same IP address, making IP similarity an infeasible measure.

Two notable events i.e. the ICDM 2015: Drawbridge Cross-Device Connection competition (Drawbridge, 2015) and the CIKM Cup 2016 Track 1: Cross-Device Entity Linking Challenge (CIKM, 2016) have resulted in many short publications where participants shared their methodologies, findings and experiences in using machine learning to conduct cross-device tracking (Díaz-Morales, 2015; Kejela and Rong, 2015; Kim et al., 2015; Selsaas et al., 2015; Walthers, 2015; Song et al., 2016; Anand

and Renov, 2015; Landry and Chong, 2015; Cao et al., 2015; Tay et al., 2015; Lian and Xie, 2016; Tran, 2016).

In the Drawbridge competition, participants are tasked to infer user identities by learning which cookies and devices belong to which individual without using PIIs. The provided dataset contains a set of training devices for which user identities are known, a set of cookies for which the user identities are partially known, and a set of test devices for which identities are unknown. Each device and cookie is also associated with an anonymised set of IP addresses which serve as hints that cookies sharing the same IP address might be from the same device. Similar to Zimmeck (2017) findings, most contestants found that IP address is the most meaningful feature to correlated devices belonging to the same user. However, they did not explore the feasibility of web domains for cross-device tracking as such information is not included in the provided dataset.

In comparison, the CIKM challenge provided participants with a collection of browsing logs belonging to anonymous users instead of cookie, device and IP information. Each browsing log contains a list of events for a specific userID and each event is mapped to a hashed URL, HTML title and timestamp. Because of the differences in data, participants of the CIKM challenge adopted an intuition that is different from that of the Drawbridge competition i.e. the browsing behaviour of a user is similar regardless of the devices they used. For example, the winner of the challenge measure the similarity of link clicking behaviour across different devices (Tay et al., 2015), while the runner up (Lian and Xie, 2016) and another participant (Tran, 2016) simply assume a user will browse the same domain, websites or webpages with similar context, even on different devices. All three participants also measure similarities in usage patterns across time, as they believe that users may exhibit some temporal patterns in their browsing behaviours. For example, some people may be more active at midnight, while some get up early in the morning. Likewise, users may also use different devices at different time, such as desktop or laptop during office hours, mobile phone during lunch time and tablets back at home in the evening.

In terms of machine learning algorithms, winner of the Drawbridge competition choose to use learning-to-rank coupled with pairwise ranking to achieve better performance over conventional binary classification. In addition, heuristically weighting gradients of cookies-pair by the reciprocal of the rank, together with cookie over-selection further improved their results. Similarly, winner of the CIKM challenge has also chosen pairwise classification over binary ones, and they utilise an unsupervised neural feature ensemble approach to learn latent features of users.

The intuition behind how two devices are related to a user is a very important consideration if we want to use machine learning for cross-device tracking. It could be based on frequent co-occurrence inside the same network like in the Drawbridge competition, or having the similar browsing behaviours like in the CIKM challenge, or both as discussed in (Zimmeck, 2017). The intuition of frequent co-occurrence

inside the same network will not be applicable in our studies as this will require the employees to connect their PMDs to the company internet network which we already assumed otherwise. If such situations are permitted, simply tracking pseudonymous cookies or information that uniquely identifies the employees mobile devices will be more straight forward and accurate (Vallina-Rodriguez et al., 2015). As for the intuition of having similar browsing behaviours, it will be applicable but there are certain challenges faced by our adversaries that will limit its effectiveness. For example, malicious third-parties will be limited by the number of first-party sites that they are embedded in and consequently, the amount of browsing events that they can collect from both the company and the employees PMDs. Similarly, network eavesdroppers are limited by the amount of browsing events that they can see in clear due to the use of HTTPS encryption. These limitations will result in gaps in the trained model which might reduce the accuracy of the prediction.

Nevertheless, the idea of using machine learning to train a model and subsequently use it to infer the likelihood that two devices belongs to the same user is a natural fit to the scenario that we are studying. For example, traffic collected from the company can be used to train a model that subsequently be used to predict if an unknown set of traffic belongs to an employee. The challenge will be in coming up with a relatively accurate intuition to correlate employees' ODs with their PMDs. For example, other than having similar web browsing behaviours, a user social network would remain the same regardless of the device they used. Likewise, similarities in travelling patterns may also suggest the same ownership. Next, we will look at some existing works that study the uniqueness of a person Internet behaviours, social networks and travelling patterns, as well as discuss their feasibility for cross-device tracking.

2.3.2 Tracking Online Behaviour

Behaviour-based tracking exploits the assumption that online behaviours of users, especially web browsing behaviours exhibit characteristic and recurrent patterns. Studies have shown that people revisits more than 50% of their visited pages and continually add new pages into this list (Tauscher and Greenberg, 1997). Furthermore, the strategies each individual undertake to navigate websites differs significantly and are strongly influence by personal habits and the type of sites visited (Obendorf et al., 2007). On top of these, automated processes such as browsers retrieving recently opened websites upon launch, RSS readers fetching updated content from a predefined set of blogs every day, or mail clients polling mail server for updated emails further proves that online communication, especially browsing behaviours can be unique or, at least, distinct.

Olejnik et al. (2014) is the first to conducted a large-scale study on the uniqueness of web browsing behaviours. In their study, they look at the browsing histories of 368,284 voluntary Internet users and observe that they are different for 69% of the users, and 97% of those users that visited at least 4 different websites could be

uniquely identified by their histories. They also observe that using browsing histories as user fingerprints is relatively stable. They reported that 38% of such fingerprints are identical over time while differing ones could still be correlated with original history contents, indicating static browsing preferences. An interesting finding is that they found that a small number of pages is sufficient to develop an unique behavioural fingerprint. The authors show that 50 web pages is sufficient to fingerprint 42% of the users which increases to 70% with 500 web pages.

In more practical investigations, Banse et al. (2012), Herrmann et al. (2013) and Kirchler et al. (2016) study the feasibility of tracking users sessions across different IP addresses using just DNS requests initiated by browsers. DNS requests reveal the domain names of the websites that the users visited and even though it is not as precise as browsing histories (in terms of context, order and timing i.e. not all visitation will initiate DNS requests), it has the added "disadvantage" of being commonly sent in clear even if the originating browsing activities are encrypted (Technical Center of Internet, 2016). This "disadvantage" allows network eavesdroppers to obtain a more complete picture of the users' browsing behaviours. The authors overcome the inherent imprecision by ignoring the exact timings and order of the requests when modelling web-browsing behaviours. To ascertain which browsing sessions having different IP addresses belong to the same user, Banse et al. (2012) and Herrmann et al. (2013) employ supervised classifiers such as Multinomial NaÔve Bayes, 1-Nearest-Neighbor and Yang's behavioral pattern mining. In comparison, Kirchler et al. (2016) choose to use unsupervised learning based an modified k-means clustering algorithm to overcome the need for labeled training sessions which are hard to obtain realistically by adversaries. The results show that supervised classifications is able to link up to 85.4% of the browsing sessions of all users on a day-to-day basis even on a realistic large-scale dataset of more than 3600 users, while unsupervised learning only managed to link up to 73% of those highly active users.

In comparison, Gu et al. (2016) combine several features such as website domain, access frequency of each domain, geographic location of the user, mail domain, search engine, user-agent, usage frequency of IM and usage frequency of P2P to represent user behaviour in a session. On top of this, they also model user preferences using product categories that the user searched for in shopping websites. To link users to their Internet sessions, the authors also employ supervised learning using Multinomial NaÔve Bayes to classify behaviours and naive Bayes classifier to classify user preferences. The class that gives the maximum value when combining the results from these two classifications will be the user of the unknown session. The authors reported that an average 93.79% of 509 active users can be identified correctly when they combine user behaviours with their preferences.

Results from these prior studies suggest that tracking user based on online behaviours is feasible. However, all of them did not investigate the performance of their technique for cross-device tracking. In a more related study conducted by Kane et al. (2009), it is reported that users tend to visit many same websites on both their

mobile devices and desktops. Their results reveal that 75.4% of the domains visited on users' mobile devices are also visited on their desktops. However, the authors also found that users visit a much higher number of websites on desktops than on their mobile devices, and thus only 13.1% of the domains visited on users' desktops were found on their mobile devices. Despite such disparity in browsing habits between desktops and mobile devices, the authors still conclude that users exhibit similar browsing behaviours across devices.

2.3.3 Tracking Social Networks

Social networking services are increasingly sharing information about users and their networks for profits, and to support application development so as to further their market dominance. Although they ensure that privacy is protected through anonymisation, i.e., removing PIIs like names, email addresses, etc., they do not prevent unknown users from being re-identified to their accounts because each user social network is unique (Narayanan and Shmatikov, 2009; Wondracek et al., 2010; Su et al., 2017). Furthermore, they usually do not anonymise the account identifier and this enables the re-identified user to be "personally" identifiable through their public profile pages.

Narayanan and Shmatikov (2009) de-anonymise users of an anonymised, sanitised target network by comparing it with a known auxiliary social network. The intuition is that both networks have some overlap in membership and structure, and these similarities can be exploited to calculate the probability that a user in the anonymised, sanitised target network also belongs to the auxiliary social network, thus revealing their identities. The authors further explain that auxiliary networks can be realistically obtained by attackers, for example, by crawling the online network or from malicious third-parties. Attackers may also directly collude with an operator of a different network whose membership overlaps, or if they are government-level actors, they may collect the data via surveillance and court-authorised searches. Depending on the type of attacker, the users in the auxiliary network may be a subset, a superset, or overlap with those of the target network. Using this technique, the authors successfully de-anonymise several thousand users in an anonymous Twitter network, using a completely different social network, Flickr, as the source of auxiliary information, even when the overlap between them is relatively small.

Likewise, Wondracek et al. (2010) ascertain that the group memberships of a user (i.e. the groups that the user belongs to in a social networking service) can uniquely identify a person, or at least, significantly reduce the number of possible candidates. The intuition is that most social networking services have features that allow users to join groups that shares similar interest, demographic and orientations etc.. As such, the set of groups where users are members of could serve as fingerprints to uniquely identify them. The authors argue that even the groups that the most active user joins only form a small fraction of all groups in the social networking service, and thus,

making groups membership distinct and unique. The authors further explain that it is possible to identify the groups that a user belongs to through their browsing histories. For example, some social networking sites will indicate the group name or id in their hyperlinks, and when the user clicks on those links, the corresponding urls will be recorded as part of the browser histories. An attacker, through History Stealing attack (Jackson et al., 2006; Jakobsson and Stamm, 2006) can then probe the user browser on whether a list of predetermined (i.e through analysis of the social networking service) group related urls are stored in the browser history. The effectiveness of this technique is demonstrated on the Xing social network and the authors show that it is able to potentially de-anonymise million of its users.

More recently, Su et al. (2017) propose that since users' social networks are distinct, the set of feeds that they received, like group memberships in (Wondracek et al., 2010), can also uniquely identify a person, or at least, significantly reduce the number of possible candidates. The intuition is that users will visit links in their feed with higher probability than a random user, and thus when coupled with the uniqueness of their feeds, makes their browsing histories distinct from other users in the same social networking service. Given a set of anonymous browsing histories, the authors demonstrate that one can identify a list of potential candidates by first checking with the social networking service for the list of users that posted those links and after that, find out who are their followers. Re-identification involves comparing the set of anonymous browsing histories with each candidate's feeds and finding the likelihood that the former belongs to the latter. The candidate with the best likelihood score is most likely the owner of that set of anonymous browsing histories. The authors test their technique on Twitter and evaluation results based on simulated dataset show that given a set of histories with 30 links, they can de-anonymise users more than 50% of the time, while evaluation results based on 374 real-life users achieved a much better 72% accuracy.

The above three researches prove that social networks can be used effectively to identify unknown users, either by comparing the social network graphs, group memberships or feeds. Although all three of them never implement or test their techniques for cross-device tracking purposes, we believe that their ideas are also applicable but would require different implementations. For example, in the scenario that we are studying, if somehow, through some techniques, the adversaries are able to identify the employees' accounts, they could obtain the employees' feeds by looking up the social networking service's publicly available dataset. After that, they could compare a set of anonymous browsing histories with the retrieved feeds and ascertain if they belong to one of the employees using the algorithm in (Su et al., 2017).

One method to identify the employees' accounts is to derive them from the browsing traffic observed from the company using the algorithm in (Su et al., 2017). As these browsing traffic are generated by the employees, a subset of the candidates will be definitely be them and adversaries will just need to filter away those that are not. One option could be finding overlapping clusters in the candidates' social network

graphs since it might be assumed that employees will form strongly connected communities with their colleagues in their social networks. Another option is to compare candidates' social networks with networks from another social networking services such as LinkedIn or Facebook that might reveal employment information using the algorithm in (Narayanan and Shmatikov, 2009).

2.3.4 Tracking Travel Patterns

Increasingly, web services are interested in obtaining users' location to provide more personalised services and advertisements. This is prevalent especially in mobile applications because the device that they are installed in tends to be mobile and usually contain Global Positioning System (GPS) sensors where very accurate geo-location information can be obtained. Therefore, if two such devices are consistently found together in the same location across multiple time durations, they could be "co-travelling" and it might be possible that they are owned by the same person. The uniqueness of geo-location data is proven in a study by (Rossi et al., 2015a) where they found that it just take two spatial data items to uniquely identify nearly all of the users in GPS trace collections.

There are many researches that exploited this notation of co-travelling to correlate users found in separate location datasets (Naini et al., 2016; Riederer et al., 2016; Ma et al., 2013). The general intuition is that location datasets corresponding to each user are expected to be similar because of similar travelling habits, but not exactly identical due to inherent randomness in user's behaviour.

Naini et al. (2016) exploit this intuition to develop an algorithm that matches the histograms of two location datasets that are gathered from two separate experiments, where users identities are anonymised in one of the dataset but known in the another.

Similarly, Riederer et al. (2016) link accounts of the same user across location datasets generated from two different domains that produce location data randomly and independently from each other. In such cases, uniqueness of a user in one dataset does not imply that they will be easily identifiable in the other. Nevertheless, relatively good matching accuracies are still achieved.

In another study, Ma et al. (2013) investigate how an adversary could infer the true identity of users, either uniquely or with high probability, by comparing their anonymised location traces using a relatively small amount of auxiliary location information. The authors argue that those auxiliary location can be realistically obtained because whereabouts in public spaces can be openly observed either through chance or deliberately engineered. They could also be inferred, though less precisely, from conversations, news articles, online social networks, or web blogs.

Applying this intuition of co-travelling to cross-device tracking, we could similarly treat the two datasets as geo-location information collected from two different devices and compute their similarities to determine if they belong to the same user. However, it might not be applicable to our study since we assume that one of the device i.e. the office computer is stationary located in the company. Furthermore, if adversaries know the geo-location of the company, they could always assume mobile devices frequently located in that geo-location belong to an employee. However, this is opportunistic as it depends on mobiles apps transmitting the geo-location information. Therefore, this technique is usually not used as the primary attack mechanism, but rather as a complement.

Nevertheless, if somehow, through some techniques, adversaries are able to identify employees' social networking accounts, we could obtain a set of known locations about the employees since most social networking services offer the possibility of associating the current location of users to their posts and photos. After that, adversaries could compare a set of anonymous locations with them and ascertain if those locations belong to one of the employees using the algorithm in (Ma et al., 2013).

This idea of using location data in social networking services to de-anonymise users is an emerging field of study. Rossi and Musolesi (2014) present three strategies to identify users in an anonymised location dataset based on their check-in locations published in Location-Based Social Network (LBSN) such as Brightkite, Gowalla and Foursquare. The first strategy is a trajectory-based approach where user are identified simply by considering the trajectory of spatio-temporal points given by his or her check-in activity. The second is a series of probabilistic Bayesian approaches where user are characterised by their check-in frequencies at each location. The frequency information are also augmented with that of the users' neighbours in the LSBN social graph. The third is a hybrid strategy that combine the trajectory-based and frequency-based techniques. The results from their experiments show that it is possible to identify users using just 10 LSBN check-in locations and generally, the best identification accuracy is achieved using the hybrid strategy. Even though this study was done on LSBN, the author suggest that it will also be applicable to social networking services that offer the possibility of associating the current location of users to their posts and photos.

Similarly, Cecaj et al. (2016) use geo-locations embedded in posts and photos from social network services to de-anonymise Call Detail Records (CDR) dataset released by mobile telecom operators. Their results confirm Rossi and Musolesi (2014) findings that a few location points are sufficient to unique identify majority of the users in the CDR.

In another related study done by Rossi et al. (2015b), the authors investigate the role of location semantics in the identification of LBSN users. More specifically, they would like to find out 1) what types of venues should adversaries monitor to maximise the success of identification, and 2) how often should users publish their

check-ins. Their analysis show that different venue types have different level of discriminative power to reveal user identity and users home residences provided the highest re-identification success. They also found that there is no correlation between how frequently users check-in their location and the ability to successfully identify them.

2.4 Alternative Cross-Device Tracking Techniques

There are other methods to obtain information about cross-device usage. One of them is to purchase or lease device graphs from cross-device tracking company such as Tapad and Drawbridge via cookie syncing (Bashir et al., 2016). In cookie-syncing, a company send its identifiers to the cross-device company who then returns a graph of devices that are identified as belonging to the same user.

Cross-device tracking companies can also work together to improve their tracking accuracy. For example, companies that relies on probabilistic methods can contract those using deterministic methods to verify the accuracy of their algorithm while those using deterministic can expand their coverage by including probabilistic matches.

Finally, there are also novels cross-device tracking approaches that does not exploit web communication, such as detecting ultrasound embedded in television content using smartphone microphone (Waddell, 2015). However, these together with the rest of the alternative cross-device tracking techniques are outside the scope of this study.

Chapter 3

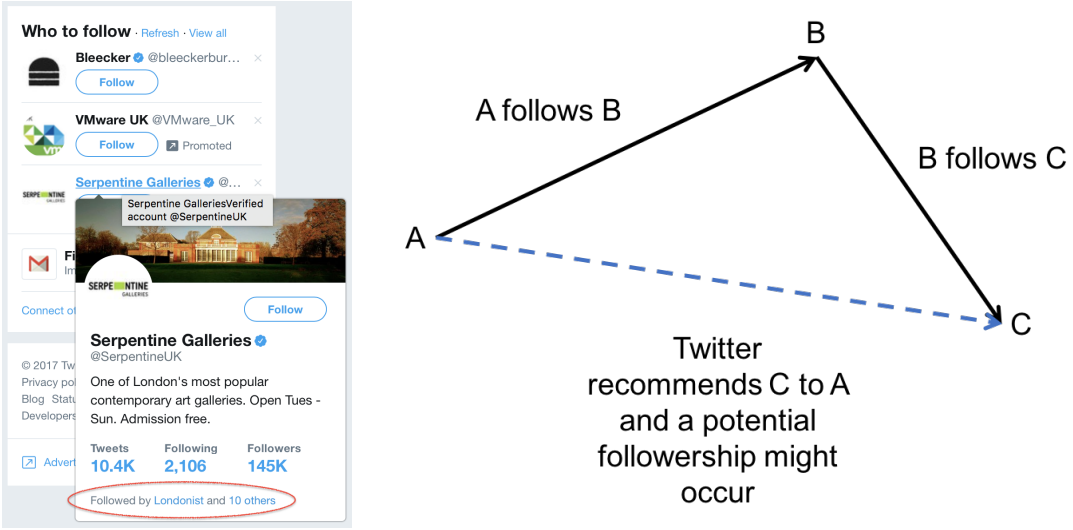
Technical Background

In this section, we review the technicalities that underpins our strategy. Firstly, we discuss how *directed triadic closure* influence who a social media user follows and how this leads to employees in a company sharing similar followees and feeds over time. We also show how such sharing will enable us to deduce the employees' accounts even when the observed URLs are a mixture coming from many different employees. Secondly, we explain in-depth how the MLE formula that we have adopted in many parts of our algorithms is derived. Finally, we introduce the Twitter APIs and review those functions that we use in our experiments.

3.1 Directed Triadic Closure

Many social media services proactively find and recommend new accounts for their users, such as the one shown in Figure 3.1(a). As circled in red in Figure 3.1(a), social media services often recommend users who are the followees of one's followees. For example in Figure 3.1(b), A and C initially do not have any relationship, but because B is a followee of A and C is a followee of B, Twitter recommends C to A and even indicate to A how many of its followees follow C. The idea is that A will be more willing to follow C when it knows that many of its followees is also following C. Recommendation of C to A could also be made by B, but we expect it to happen less often than those made by social media services who have very strong incentives to help their users build their social networks.

Romero and Kleinberg (2010) term such process of gaining followership as *directed triadic closure* and found that it plays an important role in the formation of relationship in Twitter. The authors explain that *directed triadic closure* is analogous to *triadic closure* in traditional social networks, which is the most common way friendships are formed between strangers. That is, if two strangers share one or many common friends, the chances of them becoming friends are much higher than those with no common friend. This is because these common friends or any intermediaries (i.e. those with knowledge of the common friendships) could act as go-betweens to introduce the two strangers (Rapoport, 1953; Granovetter, 1973;



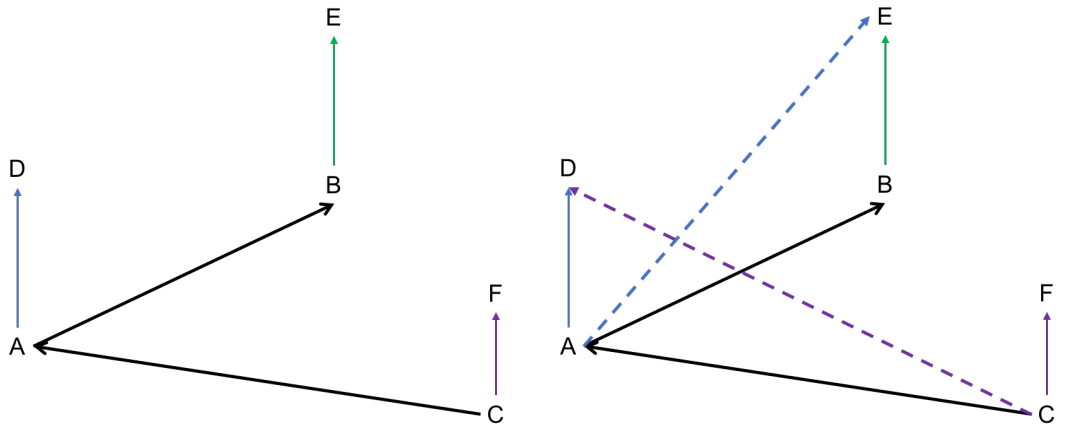
(a) How Twitter recommends new followees to users.

(b) Graphical illustration of how two strangers can establish a relationship.

Figure 3.1: How *directed triadic closure* influence the building of relationships.

Easley and Kleinberg, 2010). Likewise in social media services, common friends or a recommendation system could help establish new relationships (like what Twitter did in Figure 3.1(a)), even if their networks are primarily *directed information networks* that facilitate dissemination of media contents more than building friendships. However in such networks, the resultant relationships are usually single directional (i.e. follower-followee relationship) and remains so until the followee reciprocate the relationship, thereby creating a bi-directional, reciprocal friendship.

In this report, we further explore the effect of *directed triadic closure* at the group level and show how it leads to employees in the company sharing similar followees and feeds. To aid explanation, we illustrate the effect of *directed triadic closure* on a small group of users (A,B,C,D,E and F) in Figures 3.2,3.3 and 3.4. Similar to Figure 3.1(b), existing follower-followee relationships are depicted as arrows pointing in the direction of the followees, while potential ones are depicted in similar, but dashed arrows. In the first scenario (Figure 3.2(a)), all subjects are not members of any social group and none of them have reciprocal relationships between them. Over time, because of recommendations that are influence by *directed triadic closure*, A could potentially follows E, and C could potentially follows D as shown in Figure 3.2(b). As a result, A and B might share E as a followee, while A and C might share D as a followee.

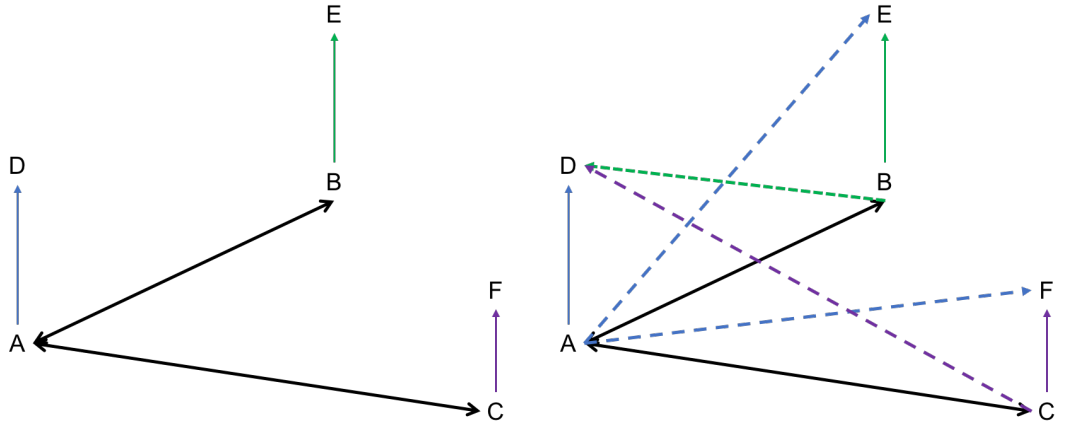


(a) Relationships between A,B,C,D,E and F before the recommendations.

(b) Relationships between A,B,C,D,E and F after the recommendations.

Figure 3.2: Scenario 1: Relationships between A,B,C,D,E and F before and after the recommendations.

In the second scenario (Figure 3.3(a)), B and C reciprocate their relationships with A. Reciprocity in a online relationship is usually a gesture of friendship, suggesting that both parties knows each other (Xie et al., 2012; Natali and Zhu, 2016; Kim et al., 2016). With reciprocity, AB and AC are now dyads, the smallest form of human social group. Similarly, over time, A could potentially follows E and F, while B and C could potentially follow D, because of the recommendations. As we can see in Figure 3.3(b), reciprocal relationships and being in a social group increase the number of followees two users could potentially share, just like how AB might now share D and E, and AC might now share D and F.

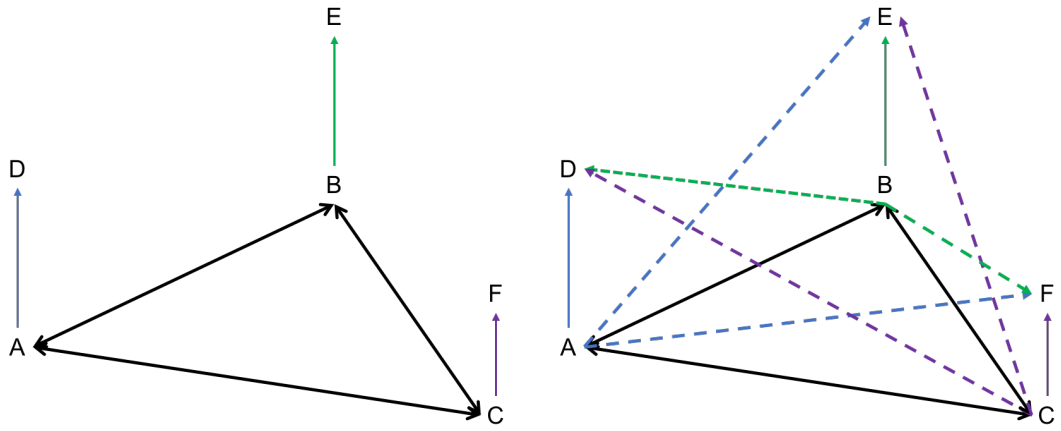


(a) Relationships between A,B,C,D,E and F before the recommendations.

(b) Relationships between A,B,C,D,E and F after the recommendations.

Figure 3.3: Scenario 2: Relationships between A,B,C,D,E and F before and after the recommendations.

Directed triadic closure would, of course, also increases the possibility of B and C forming a friendship. Therefore, in the third scenario (Figure 3.4(a)), A, B and C form a three-way reciprocal relationships with each other. ABC is now a triad social group, which some consider the building blocks of real-life social groupings since they are formed more frequently between members of the same group, rather than those from different groups (Granovetter, 1973; Easley and Kleinberg, 2010). Now, after the recommendations, A could potentially follows E and F, B could potentially follows D and F, and C could potentially follows D and E. This leads to D, E and F potentially becoming shared followees to triad ABC as depicted in Figure 3.4(b).



(a) Relationships between A,B,C,D,E and F before the recommendations.

(b) Relationships between A,B,C,D,E and F after the recommendations.

Figure 3.4: Scenario 3: Relationships between A,B,C,D,E and F before and after the recommendations.

From these three scenarios, we can see that members of a social group are likely to follow similar users over time. This is because these users will be recommended to the members as there are already existing members following them. This potentially increases the number of members following those users, and as more members join in, the more likely other members will follow suit. Therefore, when *directed triadic closure* occurs at the group level, it is highly likely that there will be a snowballing effect which results in members sharing similar followees and feeds over time. This snowballing effect is also likely to be more substantial when there are more reciprocal and triadic relationships within the group. As such, we hypothesise that each social group, over time, are likely to have a set users that are followed by many, if not all of the members. We also hypothesise that this set of followees is likely to be unique to the group especially if many reciprocal and triadic relationships exist, unless there are other groups that have similar memberships structure.

Given that the followees of a user are strongly influenced by the social groups that he or she is involved in, we can therefore group a user's followees by his or her social circles as illustrated in Figure 3.5. Although there will be some that appear in multi-

ple social circles, but for the sake of illustration, we assume that each followee only belongs in at most two social circles. This assumption will not affect the correctness of our algorithm as, we shall see later, we are only concerned if a followee belongs or do not belong to a social circle of interest.

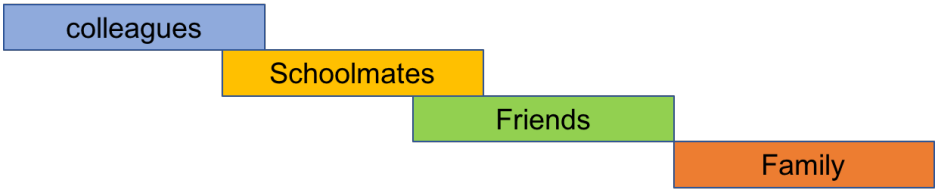


Figure 3.5: A user's followees can be grouped according to a user's social circles.

Therefore, given a set of URLs observed from a company, there might be some that are posted by these shared followees and others posted by those that are unique to individual employees (as illustrated in Figure 3.6). We assume that it is not possible to differentiate which of the URLs belong to which employees, given that they have been mixed together when they pass through the company's NAT gateway

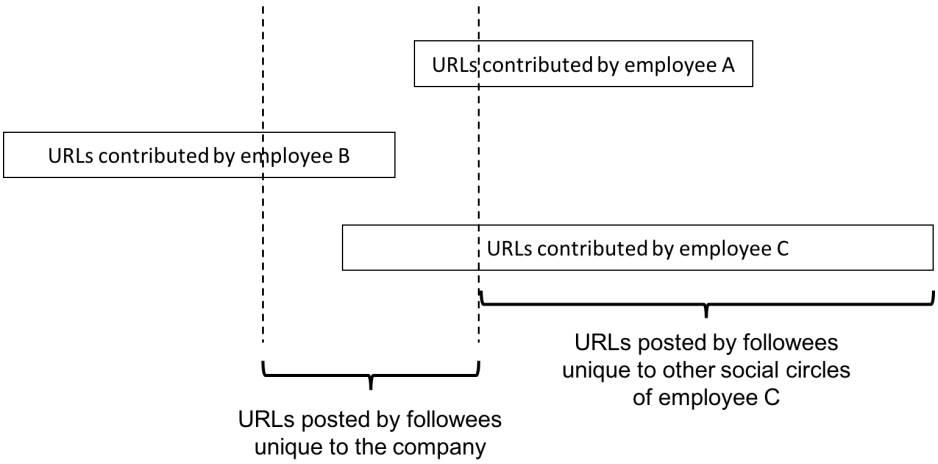
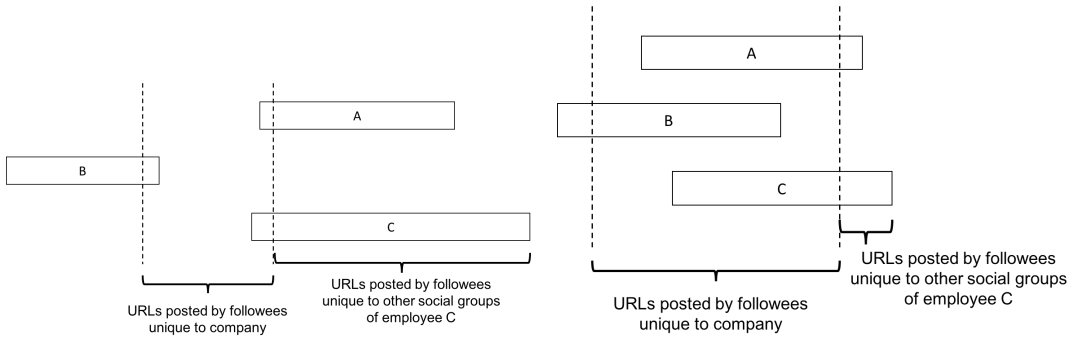


Figure 3.6: An example of how a set of URLs observed from a company could be contributed by its employees.

Since the links visited by each employee are random, there will be times where majority of the observed URLs are unique to individual employees (see Figure 3.7(a)), and at times unique to the company (see Figure 3.7(b)). As such, we assume that these two scenarios form the end cases of how a set of URLs observed from a company are contributed by its employees. If most of the URLs falls under the first scenario, there is little overlap and it will be fairly easy and accurately to deduce the employees involved, since each portion of the URLs (when sufficiently large) uniquely represent a user (Su et al., 2017). If most of the URLs falls under the second scenario where there are significant overlaps, we are still able to deduce at least one of the employees accurately because the overlapped URLs are unique to the

company. The deduced employee, however, might not have visited the most number of matching links in the observed URLs. But rather, have the most number of matching links in his or her feeds. For example, if each member A,B and C in Figure 3.7(b) just visited either one or two matching links in the observed URLs, none of them will be deduced if there is another member D whose feeds contain links corresponding to all the observed URLs.



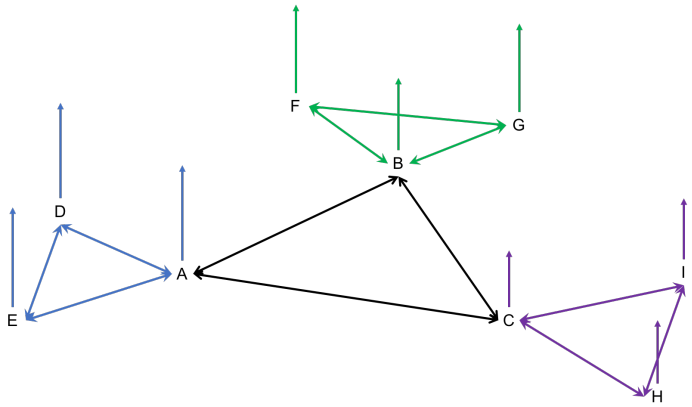
(a) End case where majority of the observed URLs are unique to individual employees.

(b) End case where majority of the observed URLs are unique to the company.

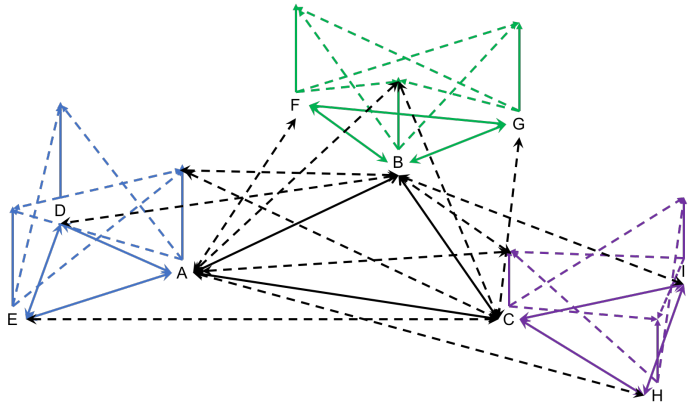
Figure 3.7: Two possible end cases of how a set of URLs observed from a company could be contributed by its employees.

Of course, only very small companies have such simplistic social network structures consisting of only one or two social groups. Large companies have multiple small, densely clustered social groups that are interconnected by weaker bridges (Granovetter, 1973; Easley and Kleinberg, 2010) like Figure 3.8(a), and after the *snowballing directed triadic closure* process, might evolve into Figure 3.8(b). URLs coming from such structures are usually an assortments of URLs that are unique to the various smaller social groups as well as to individual employees as illustrated in Figure 3.8(c). This however, will not deter us from deducing at least one of the employees, since majority of the URLs will still either be unique to one of the social groups or an individual employee.

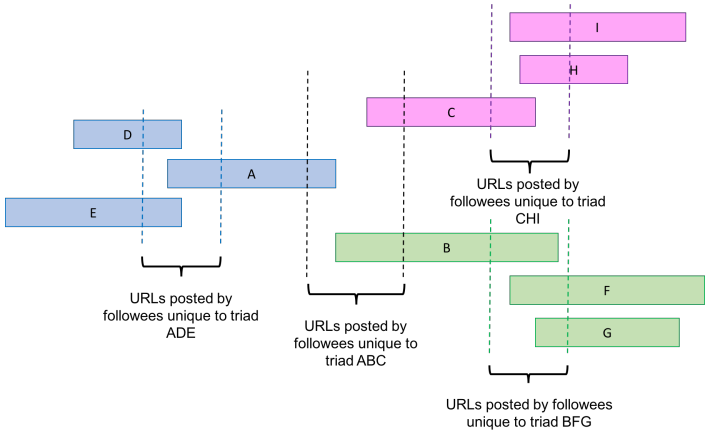
Therefore, given that in the two end cases (i.e. both for small and large companies), at least one of the employees can still be deduced from a set of URLs observed from a company. As such, we hypothesise that it will be possible deduce at least one of the employees in any other scenarios since they will always fall between these two end cases. We show how this can be achieved using our algorithm in Section 5.1.



(a) Social network structure in large companies usually consist of multiple small, densely clustered social groups interconnected together by weaker bridges.



(b) Social network structure of a large company depict in Figure 3.8(a) after the *snowballing directed triadic closure* process.



(c) One possible scenario of how a set of URLs observed from the company depict in Figure 3.8(b) could be contributed by its employees.

Figure 3.8: Social network structure of a company before and after *snowballing directed triadic closure*, and how a set of URLs observed from it could be contributed by its employees.

3.2 Maximum Likelihood Estimation

In our algorithms, we adopted the Maximum Likelihood Estimation (MLE) formula from (Su et al., 2017) to determine which set of social media feeds will most likely generate the observed URLs. We have chosen this formula because it is specially crafted to work with social media feeds and explicitly adjust for the size of a user's followees. This helps to ensure that the formula will be compatible with our algorithms and are not biased towards users with too many followees.

3.2.1 Formula's Basic Concept

Su et al. (2017) conceptualise the MLE formula as way to measure the likelihood of a user visiting a link, and is governed by two considerations: 1) the link's overall popularity (i.e. how many users embed it in their postings) and 2) whether it appears in a user's feeds. The first consideration is represented by the parameter p_R in the MLE formula and it looks at the total background probability of clicking on links found in a user's feeds. Since it is impossible to determine the exact distribution of such probabilities, Su et al. (2017) approximate them by assuming that each followee posts links with a fixed total probability, and estimate p_R as:

$$p_R = \lambda \cdot |\text{user's followees}| \tag{3.1}$$

where $\lambda = e^{-15}$. The parameter λ is also loosely estimated by Su et al. (2017) based on the volume of links that they have seen in their studies of Twitter. p_R takes into account the size of one's followees and the popularity of links that are posted by them, thereby allowing the MLE formula to adjust for the size of a user's followees.

The second consideration is represented by the parameter q_R in the MLE formula and it looks at the fraction of visited links that appears in an user's feed. Su et al. (2017) denotes q_R as:

$$q_R = \frac{|U \cap L|}{|U|} \tag{3.2}$$

where $L = \{l_1, l_2, ..., l_N\}$ is a collection of links that are embedded in the social media feeds received by a user, and $U = \{u_1, u_2, ..., u_n\}$ are the set of URLs that are observed when a user visits links embedded in social media contents. Note that U is not necessary a subset of L as users can also read social media contents that they did not receive in their feeds.

3.2.2 Formulating the Likelihood Function

To derive the MLE formula, we first need to formulate the likelihood function. To do this, we conceptualise the problem from a different perspective i.e. how likely are the generated URLs U drawn from a user's feeds L as illustrated in Figure 3.9.

This perspective better follows the standard MLE problem statement: Given a sample of n independent and identically distributed (i.i.d.) observations, coming from a distribution with a probability density function that depends on some unknown parameter θ, the goal of MLE is to find a value of θ that will maximise the joint probability density function of the observations(Wikipedia, 2017c).

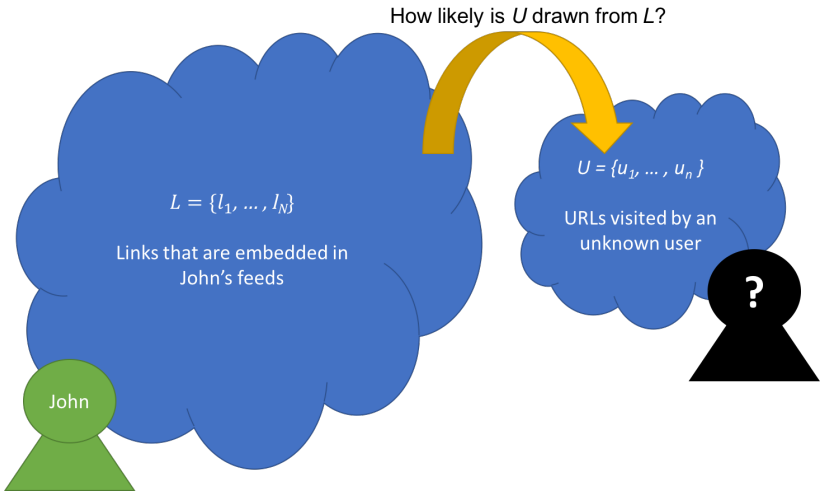


Figure 3.9: Illustration of the objective of the MLE formula discussed in (Su et al., 2017).

Therefore, in our scenario, we assume that U is the sample of n i.i.d. observations coming from the distribution L, with a probability distribution that depends on some unknown parameter θ. The probability density function of the t-th observation is therefore $f(u_t|\theta)$ and the joint probability density function for all the observations $u_1, u_2, ..., u_n$ is:

$$P(u_1 = l_1, u_2 = l_2 ..., u_n = l_N) = f(u_1|\theta) \cdot f(u_2|\theta) \cdots f(u_N|\theta) = \prod_{i=1}^{n} f(u_i|\theta) \tag{3.3}$$

When we look at this joint probability density function from a different perspective such that sample U is fixed whereas θ is the function variable, this function now becomes a likelihood function that measures how likely is sample U drawn from population L, given different values of θ:

$$\mathscr{L}(\theta) = \prod_{i=1}^{n} f(u_i|\theta) \tag{3.4}$$

Suppose that we label u_t as 1 if it matches a link in L and u_t as 0 if it does not, the probability density function of the t-th observation in sample U can be rewritten as:

$$f(u_t|\theta) = \theta^{u_t}(1-\theta)^{1-u_t} \tag{3.5}$$

and the likelihood function $\mathscr{L}(\theta)$ in Equation 3.4 as:

$$\mathscr{L}(\theta) = \prod_{i=1}^{n} \theta^{u_i}(1-\theta)^{1-u_i} \tag{3.6}$$

To determine the maximum likelihood, we need to find the value of θ that maximise the likelihood function $\mathscr{L}(\theta)$. This can be done by differentiating $\mathscr{L}(\theta)$ with respect to θ and setting the derivative to 0:

$$\frac{\partial \mathscr{L}(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta}[\prod_{i=1}^{n} \theta^{u_i}(1-\theta)^{1-u_i}] = 0 \tag{3.7}$$

However, this operation is usually hard to work with. To make the differentiation easier, the natural logarithm of the likelihood function are often used instead, since multiplication and power operations can be simplified using logarithm:

$$\log ab = \log a + \log b \tag{3.8}$$

$$\log a^b = b \log a \tag{3.9}$$

Moreover, taking the natural logarithm do not affect maximisation because natural logarithm is a increasing function (see Figure 3.10), that is if $x_1 < x_2$, then $f(x_1) < f(x_2)$. This means that the value of θ that maximise the natural logarithm of the likelihood function $\ln(\mathscr{L}(\theta))$, is reflective of the value of θ that maximise the likelihood function $\mathscr{L}(\theta)$,. Throughout the remainder of this report, we use ***log*** to denote natural logarithm.

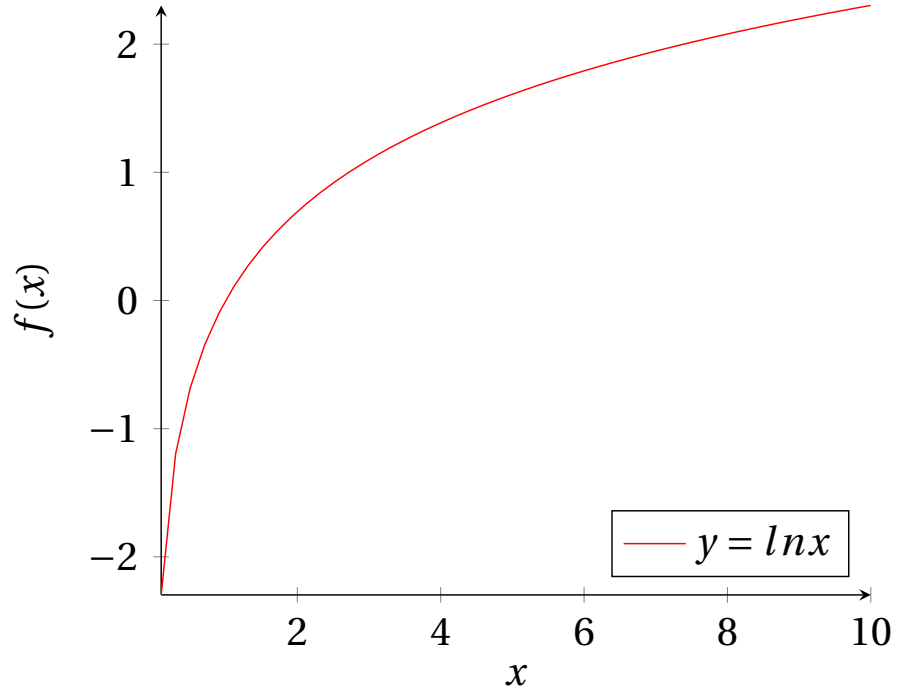


Figure 3.10: natural logarithm is a increasing function

Taking the natural logarithm of Equation 3.6 and simplify it using Equation 3.8 and 3.9 gives:

$$\begin{aligned}\log(\mathscr{L}(\theta)) &= \ell(\theta) \\ &= \log(\prod_{i=1}^{n} \theta^{u_i}(1-\theta)^{1-u_i}) \\ &= \sum_{i=1}^{n} \log[\,\theta^{u_i}(1-\theta)^{1-u_i}\,] \\ &= \sum_{i=1}^{n} [\,u_i \log\theta + (1-u_i)\log(1-\theta)\,]\end{aligned} \tag{3.10}$$

3.2.3 Contextualising the Likelihood Function

Equation 3.10 is a generic likelihood function. To use it for measuring the likelihood of a user visiting a link in his or her feeds, we follow the steps cited in (Su et al., 2017) to derive the final version of the MLE formula:

Firstly, to contextualise the formula, Su et al. (2017) define θ as:

$$\theta = \begin{cases} r\,p_i/z, & \text{if } u_i \in L, \text{ and} \\ p_i/z, & \text{if } u_i \notin L \end{cases} \tag{3.11}$$

where p_i is the probability of u_i appearing as a embedded link in a social media posting (i.e. $p_i \geq 0$ and $\sum p_i = 1$), and z is a normalising factor where $z = r\sum_{u_i \in L} p_i + \sum_{u_i \notin L} p_i$.

Replacing θ in Equation 3.10 with these set of conditions gives:

$$\begin{aligned}\ell(r) &= \sum_{i=1}^{n} [\,u_i \log(r\,p_i/z) + (1-u_i)\log(p_i/z)\,] \\ &= \sum_{i=1}^{n} [\,u_i \log r + u_i \log p_i - u_i \log z + \log p_i - \log z - u_i \log p_i + u_i \log z] \\ &= \sum_{i=1}^{n} [\,u_i \log r + \log p_i - \log z] \\ &= \log r \sum_{i=1}^{n} u_i - n \log z + \sum_{i=1}^{n} \log p_i\end{aligned} \tag{3.12}$$

Since $\sum_{u_i \in L} p_i + \sum_{u_i \notin L} p_i = 1$, z can be simplified as:

$$\begin{aligned} z &= r \sum_{u_i \in L} p_i + \sum_{u_i \notin L} p_i \\ &= r \sum_{u_i \in L} p_i + 1 - \sum_{u_i \in L} p_i \\ &= (r-1) \sum_{u_i \in L} p_i + 1 \end{aligned} \tag{3.13}$$

And as $\sum_{u_i \in L} p_i$ can be understood as the total probability of clicked links appearing in one's feeds, which is the same as what parameter p_R denotes, Equation 3.13 can be further simplified as:

$$z = (r-1)p_R + 1 \tag{3.14}$$

Substituting Equation 3.14 into Equation 3.12 gives:

$$\ell(r) = \log r \sum_{i=1}^{n} u_i - n \log((r-1)p_R + 1) + \sum_{i=1}^{n} \log p_i \tag{3.15}$$

Likewise, as $\sum_{i=1}^{n} u_i$ can be understood as the total number of clicked links that are found in one's feed, which is equivalent to $n \cdot q_R$, Equation 3.15 can be simplified as:

$$\ell(r) = n \cdot q_R \log r - n \log((r-1)p_R + 1) + \sum_{i=1}^{n} \log p_i \tag{3.16}$$

3.2.4 Maximising the Likelihood Function

Next, Su et al. (2017) finds the value of r that maximise the likelihood function $\ell(r)$ by differentiating Equation 3.16 with respect to r,

$$\frac{\partial}{\partial r} \ell(r) = \frac{n\overline{u}}{r} - \frac{np_R}{(r-1)p_R + 1} \tag{3.17}$$

and setting it to 0:

$$\begin{aligned} \frac{n \cdot q_R}{r} - \frac{np_R}{(r-1)p_R + 1} &= 0 \\ \frac{q_R}{r} &= \frac{p_R}{(r-1)p_R + 1} \\ q_R(r-1)p_R + q_R &= p_R r \\ q_R p_R r - p_R r &= q_R p_R - q_R \\ \widehat{r} &= \frac{q_R}{p_R} \cdot \frac{1 - p_R}{1 - q_R} \end{aligned} \tag{3.18}$$

Replacing r in Equation 3.14 with the maximised $\hat{r}$ gives:

$$\begin{aligned}\hat{z} &= (\hat{r}-1)\,p_R + 1 \\ &= p_R \cdot \frac{\overline{u}}{p_R} \cdot \frac{1-p_R}{1-\overline{u}} - p_R + 1 \\ &= \frac{\overline{u}(1-p_R) - p_R(1-\overline{u}) + (1-\overline{u})}{1-\overline{u}} \\ &= \frac{1-p_R}{1-\overline{u}}\end{aligned} \tag{3.19}$$

Finally, substituting the maximised $\hat{r}$ and $\hat{z}$ back into Equation 3.16 gives:

$$\begin{aligned}\hat{\ell} &= n \cdot q_R \log(\frac{q_R}{p_R} \cdot \frac{1-p_R}{1-q_R}) - n\log(\frac{1-p_R}{1-q_R}) + \sum_{i=1}^{n} \log p_i \\ &= n \cdot q_R \log(\frac{q_R}{p_R}) + n \cdot q_R \log(\frac{1-p_R}{1-q_R}) - n\log(\frac{1-p_R}{1-q_R}) + \sum_{i=1}^{n} \log p_i \\ &= n \cdot q_R \log(\frac{q_R}{p_R}) - n \cdot q_R \log(\frac{1-q_R}{1-p_R}) + n\log(\frac{1-q_R}{1-p_R}) + \sum_{i=1}^{n} \log p_i \\ &= n \cdot q_R \log(\frac{q_R}{p_R}) + n(1-q_R) \log(\frac{1-q_R}{1-p_R}) + \sum_{i=1}^{n} \log p_i\end{aligned} \tag{3.20}$$

Since $\sum_{i=1}^{n} \log p_i$ and n are fixed constants, they can be ignored when comparing the MLE scores across different users. Therefore, the MLE formula can be further simplified as:

$$\hat{\ell} = q_R \log(\frac{q_R}{p_R}) + (1-q_R) \log(\frac{1-q_R}{1-p_R}) \tag{3.21}$$

where,

$$q_R = \frac{|U \cap L|}{|U|} \tag{3.22}$$

$$p_R = |\text{user's followees}| \cdot e^{-15} \tag{3.23}$$

In Sections 5.1 and 5.2, we discuss how we apply Equations 3.21, 3.22 and 3.23 in our algorithms to deduce the social media accounts of employees in a company, and verify if an anonymous PMD belongs to one of the employees.

3.3 Twitter

We implement and evaluate our strategy using Twitter, a popular online social media service where users post and interact with Tweets (i.e. messages) that are restricted to 140 characters. Like (Su et al., 2017), we have chosen to use Twitter in our implementation because Twitter links are automatically wrapped in a proprietary shortener that could be easily identified by both network eavesdroppers and malicious third-party sites in the observed URLs, most Twitter activities are public and it has open-source APIs that we can use to retrieve information that are required by our strategy.

3.3.1 Twitter Proprietary Link Shortener

For all links that are embedded in Tweets, Twitter will automatically wrap them using their own proprietary link shortening service **https://t.co** (Twitter, 2017e). By shortening the links, it allows users to share long urls in Tweets while maintaining the 140 character limit. Links are also checked against potentially dangerous sites during the conversion so as to protect its users from malicious sites that engage in spreading malware, phishing attacks, and other harmful activities. In addition, having a proprietary service allows Twitter to track links usage (e.g how many times a link has been clicked) which helps them to offer users useful insights to the engagement of their Tweets.

The **https://t.co** shortened links are fully qualified HTTPS urls and anyone with those links will be able to navigate to the destination webpages. However, these shortened links are more than just unique to the origin url, they are also unique to the Twitter user that wraps the link. A simple test shows that the same origin url produces different shortened links when it is wrapped by different users, but produces the same shortened link when it is wrapped by the same user multiple times. Therefore, in our implementations, we use these shortened links instead of the origin url to increase precision and reduce computational resources: Each origin url could be posted by many users that are irrelevant to the one generating the observed URL, and thus using the unique shortened link will leads us to only those that are relevant. Throughout the remainder of this report, we use the term *t.coURL* to refer to these Twitter shortened urls.

If an adversary is also exploiting Twitter in its implementations, it can look out for the *t.coURLs* in its observed URLs. For network eavesdroppers, they can search for them in the **referer** headers, while malicious third-party sites can examine the **document.referrer** property in the first-party webpages that they are embedded in.

3.3.2 Twitter APIs

Applications that requires interaction with Twitter can do so using the its open-source APIs. Among the different APIs offered by Twitter, only the RESTful APIs and Streaming APIs are relevant to our strategy. However, we only use the RESTful APIs in our implementations since we are only using them for experimentation. But, adversaries can optionally choose to use Streaming APIs in some part of their implementations if real-time access is required (e.g. constantly receiving updated users' Tweets in real-time). We will highlight the part where Streaming APIs could be implemented in Chapter 5, when we go into the details of our implementations.

RESTful APIs

Twitter provides two types of RESTful APIs. The first is a set of comprehensive REST API methods that allow applications to read and write core Twitter data as well as create new Tweets, read user profile, and more. The second is a Search API method for applications to search for recent or popular public Tweets.

In our implementations, we are using both the REST API and Search API methods to 1) derive Twitter accounts of potential employees using URLs that are observed from a company, and 2) collect the feeds received by each known employee and use them to verify the ownership of an anonymous PMD. Table 3.1 shows the list of API methods that we used in our implementations and we will discuss how we use each of them in Chapter 5.

No.	Method	Function	Rate-limit
1	*statuses/user_ timeline*	method to retrieve Tweets posted by a user	1500 requests / window
2	*followers/ids*	method to retrieve the list of followers' ids that are following a user	15 requests / window
3	*friends/ids*	method to retrieve the list of followees' ids that a user is following	15 requests / window
4	*users/show*	method to retrieve a variety of information about a user	900 requests / window
5	*search/tweets*	Search API method to retrieve a collection of relevant Tweets matching a specified query	450 requests / window

Table 3.1: Twitter REST API methods used in our implementations.

Streaming APIs

The Streaming APIs gives low latency access to Twitter's global stream of Tweet data. Applications using the Streaming APIs will be pushed messages related to the most recent Tweets or other events that have occurred, without any of the overhead associated with polling a REST server. However, this requires the applications to keep persistent connections to the Stream server and as such, only able to handle limited number of requests at any one time. Streaming APIs are useful for applications that require information in real-time, but since we are conducting offline experiments, we did not use them in our implementations. Nevertheless, for adversaries who are engaging in online attacks or wanted to receive updated information in real-time, they can use this instead of the REST APIs in some parts of the implementations, which we will show later in Section 5.2.

Authorisation

Applications that want to make use of both the RESTful and Streaming APIs need to first perform OAuth authentications, either on behalf of the application or a user. The former is sufficient when the application just need to make Application-only requests that do not require a user context, such as pulling Tweets posted by any user, accessing their friends and followers, or searching for Tweets using the Search API etc.. However, the application is not able to post Tweets on user behalf, use the Streaming APIs, search for users, or access direct messages between users etc. Nevertheless, applications can still perform OAuth authentications on behalf of their developers' own Twitter accounts if they want to use the Streaming APIs to take advantage of low latency access. As we assume that the adversaries are not going or able to deceive users in authorising their applications to send requests on their behalf, we therefore only make use of Application-only requests in our implementations. If adversaries want to use the Streaming APIs, they can authenticate using their own Twitter accounts.

3.3.3 Limitations

Both the RESTful and Streaming APIs has their own limitations. In particular, RESTful APIs are rate-limited i.e. Twitter limits the number of requests that will be fulfilled within each 15 minute interval. Table 3.1 shows the current of rate limits impose by Twitter for each of the API methods that we are using. These rate-limits, especially the ones impose on the retrieval of followees and follower ids, has prevented us from conducting a more comprehensive experimentation that includes a substantial number of test subjects.

We are also faced with the limitations of the Search API which only allow us to search against a sampling of recent Tweets published in the past 7 days. According to Twitter, this API is not meant to be an exhaustive source of Tweets, thus not all

Tweets will be indexed or made available via this interface. As such, we narrowed the scope of our evaluation to experiment with Tweets posted only in the last 5 days, as well as taking into consideration potential gaps in the datasets.

The last limitation that we faced is the lack of a Application-only request to retrieve the feeds received by a user. Although this function is provided by the "*statuses/mentions_ timeline*" REST API method, it requires the application to authenticate on behalf of the user that we are interested in. We work around this limitation by first retrieving the list of users that a user we are interested in is following, and subsequently retrieving the Tweets that those users posted. However, this two steps approach results in severe time penalties due to the rate limit imposed on the retrieval of followees ids.

Although these limitations reduce the scope and coverage of our experiments, they do not affect the performance of our strategy and results from the experiments are still relevant to our evaluation. Furthermore, we do not foresee them restricting the adversaries whom we expect to be more resourceful than us and are more targeted in their usage.

Chapter 4

Threat Models

In this section, we describe the adversaries that are able to track employees' PMDs using our strategy. Since our strategy requires an adversary to have access to employees' web browsing traffic (both from their ODs and PMDs), we foresee that adversaries such as network eavesdroppers and malicious third-party sites will be able to exploit our strategy as they possess the capabilities to monitor such traffic en masse.

4.1 Network Eavesdroppers

Network Eavesdroppers are adversaries that possess the ability to passively sniff and collect network traffic such as web browsing communication. Powerful network eavesdroppers such as state actors even have access to Internet backbone servers, potentially allowing them to eavesdrop on the web activities of billions of Internet users (Englehardt et al., 2015). In our studies, we assume that an adversary could be one such powerful network eavesdropper, since only they have the capabilities to possibly monitor both the employee's ODs and their PMDs. For example, a powerful network eavesdropper could monitor employees' ODs from fixed-line broadband networks (see Figure 4.1) and PMDs from mobile broadband networks (see Figure 4.2).

However, even powerful network eavesdroppers do not have the full view of each person's web browsing traffic as they are not able to see the content of end-to-end encrypted traffic. In our studies, we assume that although the adversary is powerful enough to monitor all web browsing traffic from both employees' ODs and PMDs, they cannot compromise those that are encrypted with HTTPS. Therefore, they will not be able to use our strategy with those URLs to conduct their attack. In Section 6.5, we will evaluate the efficacy of our strategy using only URLs that are unencrypted i.e. those that use HTTP.

We further assume that such an adversary is able to determine if the observed unencrypted URLs comes from a company of interest just by looking at the source IP addresses. This is possible because companies commonly lease static IP addresses from ISPs for their fixed-line broadband services. However, these IP addresses will

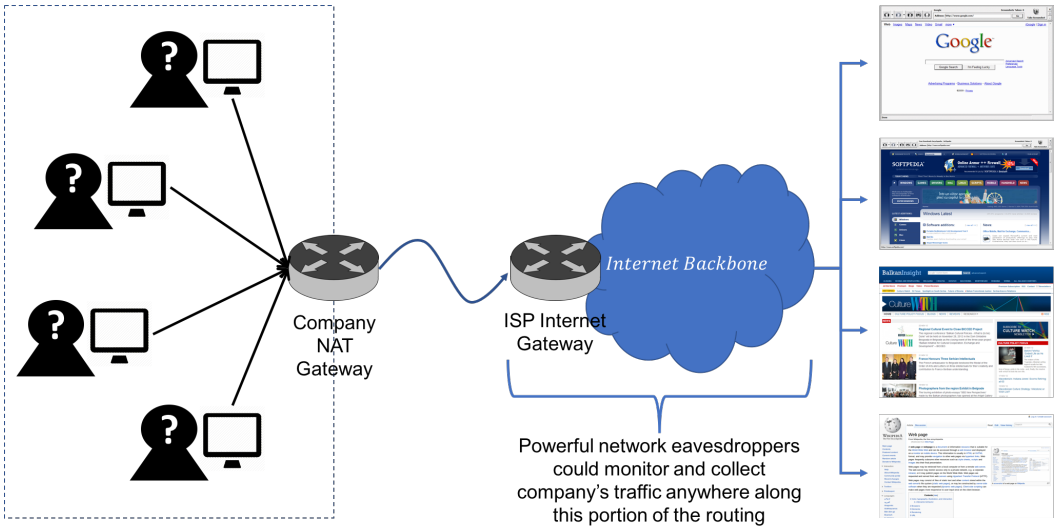


Figure 4.1: Monitoring capabilities of a powerful network eavesdropper on fixed-line broadband.

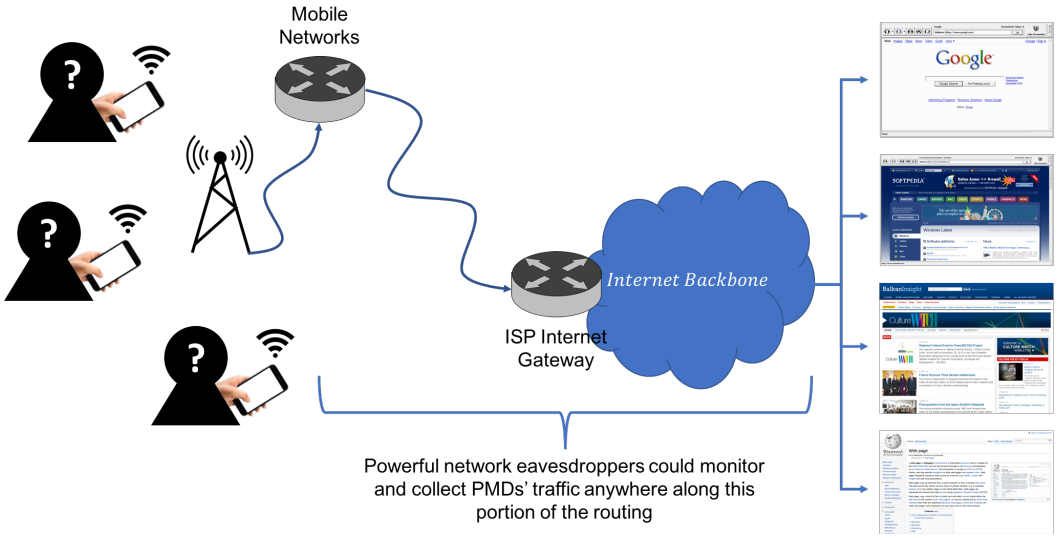


Figure 4.2: Monitoring capabilities of a powerful network eavesdropper on mobile broadband.

most likely be shared by many employees via the company's NAT gateway as illustrated in Figure 4.1. On the other hand, we assume that employees' PMDs lease dynamic, but unique IP addresses from ISPs for their mobile broadband services, which changes once the lease expire. The duration of the lease is largely ISP dependent but a large-scale study shows that fewer than 2% of clients use more than two IP addresses in a week and only 8% of clients use more than three in a month (Casado and Freedman, 2007).

We also assume that if network eavesdroppers are using our strategy with Twitter, they can increase precision and reduce computational resources by exploiting *t.coURLs* found in the unencrypted URLs that they observed. (as previously discussed in Section 3.3).

Finally, upon successful tracking of employees' PMDs, we assume that network eavesdroppers are able to compromise the PMDs using Man-In-The-Middle (MITM) or Man-On-The-Side (MOTS) attacks such as China's Great Cannon (Marczak et al., 2015) or NSA's Quantum Insert attack (Lennarthaagsma, 2015). Thereafter, using them to launch indirect attacks on the company and its networks. Even if they are not successful in tracking employees' PMDs, they can still launch spear-phishing attacks to compromise the PMDs using Twitter accounts deduced from employees' ODs, by following the technique in (Seymour and Tully, 2016).

4.2 Malicious Third-party Sites

Third-party sites are websites that have entities such as links or scripts embedded in websites that are owned by other operators. They are referred to as third-parties while the websites that they are embedded in are first-parties because the originating communication is an engagement between the user and the operators of first-party websites. When the user's browser loads the first-party websites, the embedded entities will also be loaded causing the browser to establish communications with the third-party sites. Third-party sites serve many different purposes such a payments, advertisements or analytics etc., but could also be operated by those with malicious intents such as masquerading surveillance as analytics services. Powerful third-party sites are those that have embedded links or scripts in a wide coverage of first-party websites and in our studies, we assume that an adversary could also be one such powerful third-party site.

Like powerful network eavesdroppers, powerful third-party sites also do not have the full view of each person's web browsing traffic since they will need to convince many first-party websites to embed their entities. However, they have the advantage of being able to see end-to-end encrypted traffic since the user's browser establishes communications with them. In Section 6.5, we will evaluate the efficacy of our strategy if it is being used by one of the four big third-party trackers (i.e. Google, Facebook, ComScore, and AppNexus) who have the capabilities that are reflective of

powerful third-party sites.

Likewise, we assume that powerful third-party sites know the public IP addresses of the company of interest and are able to identify URLs coming from it. But they are not able to differentiate between those coming from different employees due to NAT. Similarly, on the other hand, they are able to differentiate URLs coming from different PMDs. See Figure 4.3 for illustration of these two scenarios.

We also assume that if malicious third-party sites are using our strategy with Twitter, they can increase precision and reduce computational resources by exploiting *t.coURLs* found in the **document.referrer** property of the first-party webpages that they are embedded in (as previously discussed in Section 3.3).

Finally, upon successful tracking of employees' PMDs, we also assume that malicious third-party sites are able to compromise the PMDs using drive-by download attacks (Cova et al., 2010). Thereafter, using them to launch indirect attacks on the company and its networks. Same as network eavesdroppers, they can also launch spear-phishing attacks to compromise employees' PMDs using the deduced Twitter accounts (Seymour and Tully, 2016), if they are not successful in tracking those PMDs .

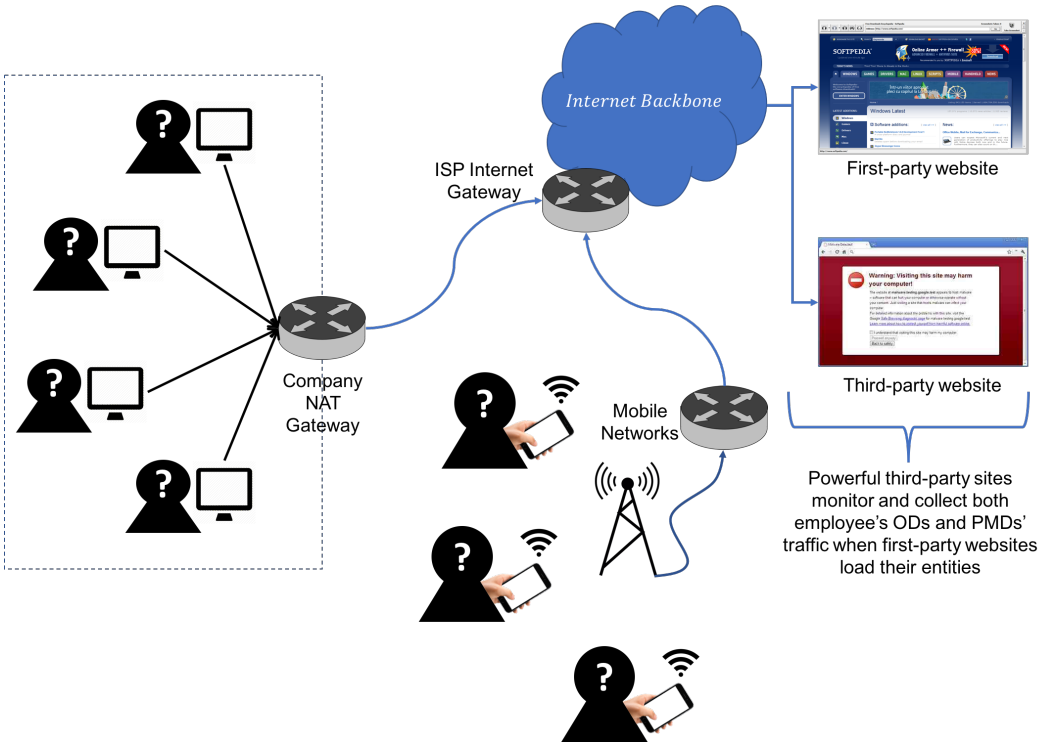


Figure 4.3: Monitoring capabilities of a powerful third-party site.

Chapter 5

Implementation

The cross-device tracking strategy that we have devised consists of two stages: In the first stage, we want to deduce employees' Twitter accounts using URLs observed from the company, and subsequently in the second stage, verify if an anonymous PMD belongs to one of the deduced employees using URLs observed from that PMD. In this section, we explain in details the algorithms that we have designed for these two stages.

5.1 Deducing Employees' Twitter Accounts

Our algorithm designed for deducing employees' Twitter accounts using URLs observed from the company proceeds in four steps: First, given a set of URLs observed from a company of interest, we extract all available *t.coURLs* from them (as previously discussed in Section 3.3) and discard those that are not relevant (see Figure 5.1).

Next, we make a series of requests to Twitter to find potential users that could have visited links that would generated the corresponding observed URLs. The underlying assumption is that the followers of a user who posted a Tweet containing a link, are more likely to visit that link than non-followers (Su et al., 2017). This is because that link will appear in the followers' feeds (which raises their awareness) and the originator is someone that they are likely to be interested in (a motivation that initiated the follower-followee relationships). Therefore, it is very likely that the user who have generated the observed URLs is a follower of the users who have posted Tweets containing the corresponding links as illustrated in Figure 5.2.

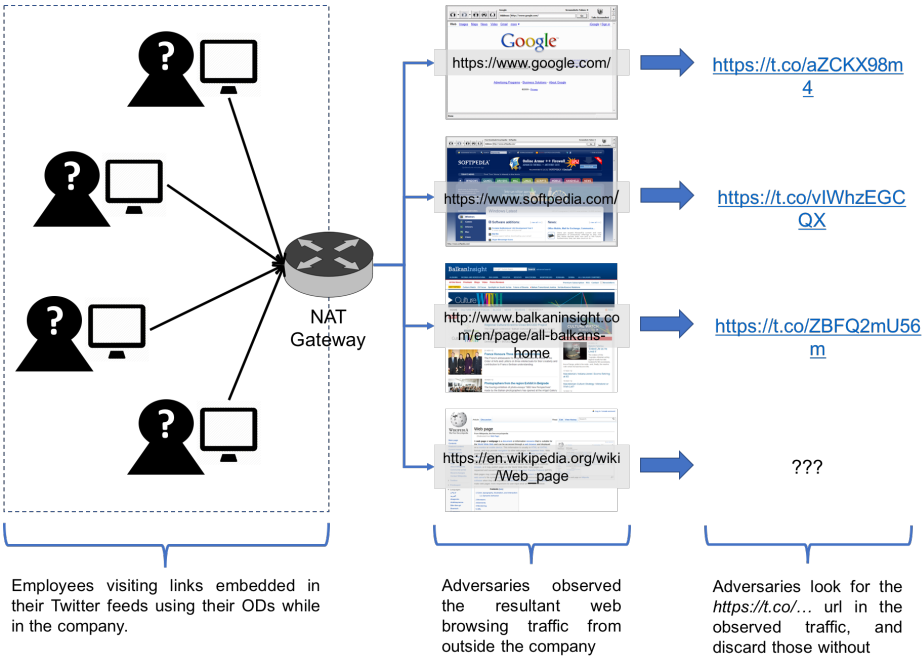


Figure 5.1: Extracting *t.coURLs* from URLs observed from a company.

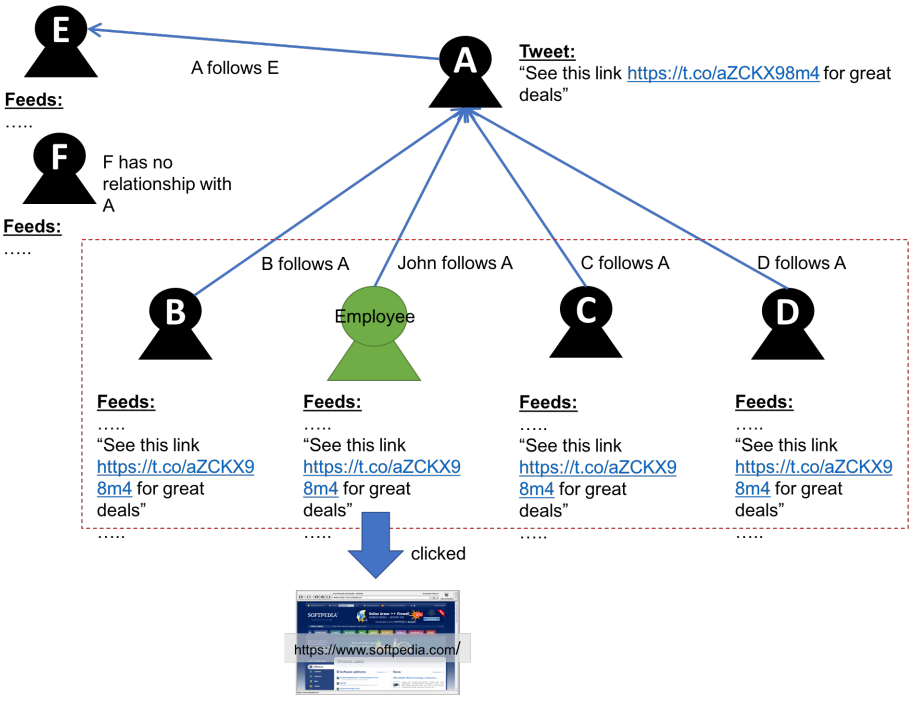


Figure 5.2: The user who have generated the observed URLs is likely to be a follower of the users who have posted Tweets containing the corresponding links.

To identify this set of followers, we follow the steps illustrated in Figure 5.3. First, (1) we search for Tweets containing one of the *t.coURLs* extracted from the observed URLs using the "*search/Tweets*" API method. After which, (2) we identify the users

who have posted those Tweets. For this step, we do not need to send requests to Twitter as the returned Tweet objects from the previous request already contain information about their posters. Using the identity of the posters, (3) we then retrieve their followers using the "*followers/ids*" API method. These three steps are repeated for all the extracted *t.coURLs*. Since it is possible that some of the employees' Twitter accounts might already be known to the adversaries (either found previously using this same algorithm or through some other means), we therefore add an optional step to (4) remove users that are known employees from the derived list of followers to increase the chances of unknown employees being identified.

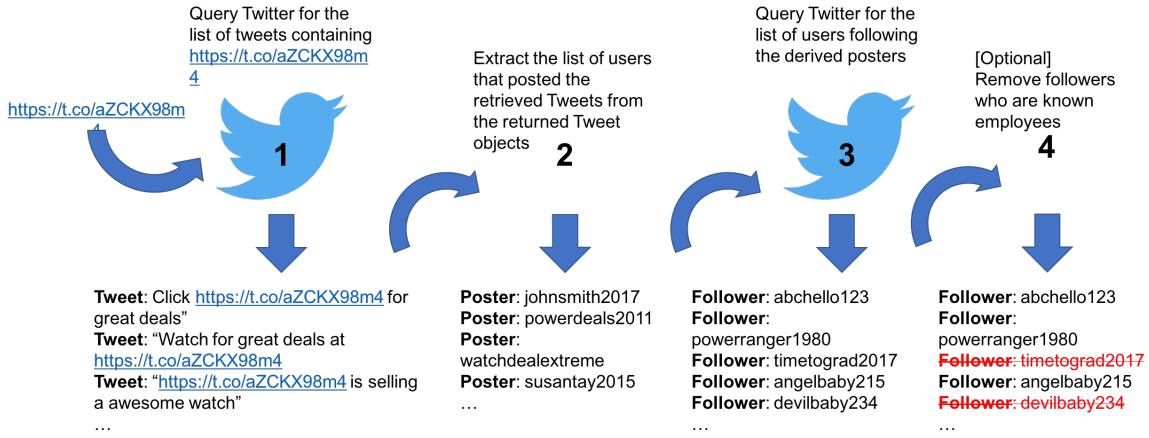


Figure 5.3: Making consecutive requests to Twitter to find users that potentially could be one of the employees.

In the third step, we again send requests to Twitter to retrieve the feeds of each derived follower (as shown in Figure 5.4). As previously discussed in Section 3.3, Twitter APIs restricts us from retrieving a user's feeds directly. Therefore, a two-steps approach is taken to overcome this limitation: First, (5) we find out who each derived follower follows (using the "*friends/ids*" API method) and then (6) retrieve the Tweets that these followees posted (using the "*statuses/user_timeline*" API method), .

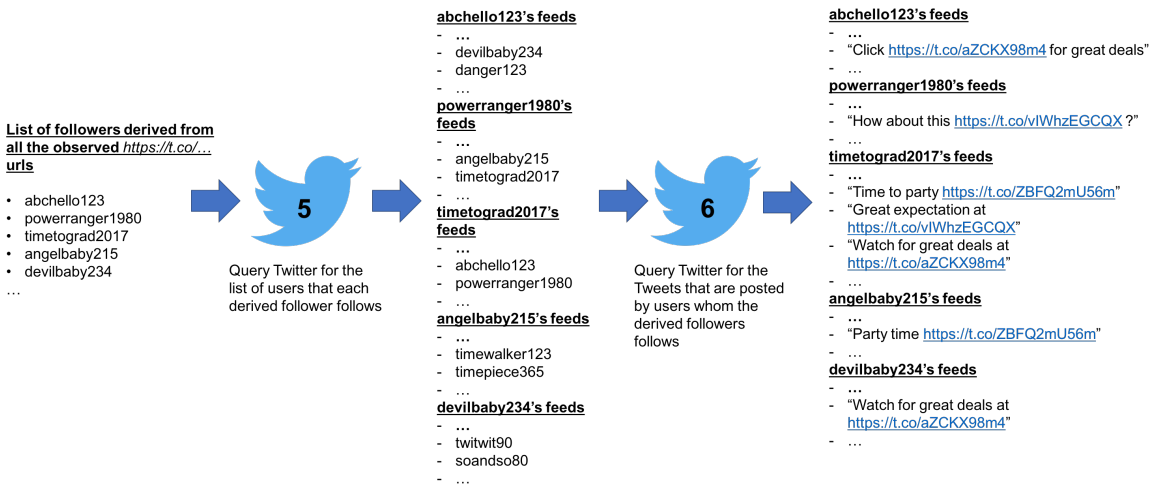


Figure 5.4: Retrieving the feeds for each of the follower that is derived from the observed *t.coURLs*.

5.1.1 Deducing the Most Likely User

In the fourth step, we find the user, from the list of derived followers, that is most likely an employee using the MLE formula previously discussed in Section 3.2. However, unlike (Su et al., 2017), instead of finding the likelihood that the *t.coURLs* in the observed URLs are drawn from the set of *t.coURLs* found in the feeds of each derived follower, we find the likelihood that a subset of the *t.coURLs* in the feeds of each derived follower are found in the set of *t.coURLs* extracted from the observed URLs as illustrated in Figure 5.5.

Figure 5.5: Finding the likelihood that a subset of the *t.coURLs* in the feeds of a derived follower can be found in the set of *t.coURLs* extracted from the observed URLs.

Recalling Equations 3.21, 3.22 and 3.23,

$$\widehat{\ell} = q_R \log(\frac{q_R}{p_R}) + (1 - q_R) \log(\frac{1 - q_R}{1 - p_R})$$

where,

$$q_R = \frac{|U \cap L|}{|U|}$$

$$p_R = |\text{user's followees}| \cdot e^{-15}$$

We can therefore calculate the likelihood score for each derived followers by setting:

- $U \equiv$ (*t.coURLs* in observed URLs) $\cap$ (*t.coURLs* in user's feeds)
- $L \equiv$ *t.coURLs* in observed traffic

and deduce that the Twitter account of an employee is the derived follower with the highest likelihood score:

$$employee \equiv \widehat{\ell}_{max} \equiv \underset{user \in \text{derived followers}}{\text{argmax}} [\, q_R \log(\frac{q_R}{p_R}) + (1 - q_R) \log(\frac{1 - q_R}{1 - p_R}) \,] \tag{5.1}$$

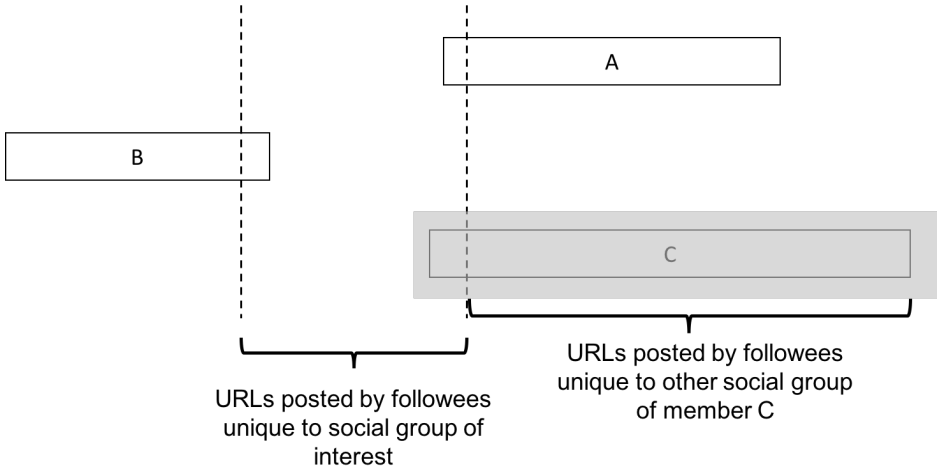
Throughout the remainder of this report, we denote this algorithm as the *Top Likelihood Deduction* algorithm. As previously discussed in 3.1, the deduced employee might not be the one that have generated the most number of observed URLs, but could be one that follows many users that have posted Tweets with links that correspond to many of the observed URLs.

5.1.2 Deducing Other Likely Users

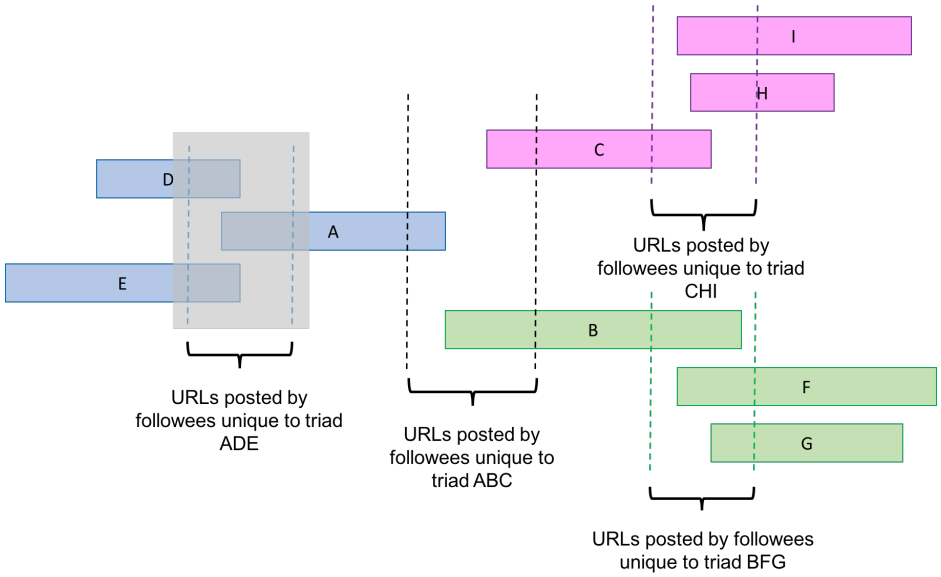
As it may be possible that there are more than one employee or one social group that may be contributing to the observed URLs (as illustrated previously in Figures 3.7 and 3.8), we therefore design another algorithm to deduce more possible employees from the observed URLs.

This algorithm first remove any user that have already been deduced from the list of followers, as well as his or her *t.coURLs* from the observed URLs. After the elimination, we repeat Section 5.1.1 and find the derived follower with the highest likelihood score again. Throughout the remainder of this report, we denote this algorithm as the *Eliminate & Iterate Likelihood Deduction* algorithm. In cases where most of the removed *t.coURLs* are unique only to an employee like in Figure 3.7(a), this will just simply remove his or her participation from the ranking of likelihood scores in the next iteration so that other employees stand a chance to compete for the top spot as illustrated in Figure 5.6(a). However, in more complex cases like Figure 3.7(a) where the most of the removed *t.coURLs* are unique to a social group in the company, this will significantly reduce the members' chances of topping the rankings of likelihood scores in the next iteration. This means that the members now can only

compete based on the URLs that they have generated and are unique to them as illustrated in Figure 5.6(b). That is, if they are not members of other social groups in the company.



(a) Scenario where most of the removed *t.coURLs* are unique only to an employee. Elimination removes his or her participation from the ranking of likelihood scores in the next iteration



(b) Scenario where the most of the removed *t.coURLs* are unique to a social group in the company. Elimination reduce the members' chances of topping the rankings of likelihood scores in the next iteration.

Figure 5.6: Different possible scenarios when removing the observed *t.coURLs*.

Nevertheless, regardless of the situations, we should still be able to deduce more employees after repeating the above elimination and ranking through several iterations until the remaining *t.coURLs* are not sufficient to calculate a set of scores. We term this lower bound threshold as *minURLs* and the higher the threshold, the lower the possibility of deducing a non-employee from the list of derived followers. However, given that we have limited resources to conducted our experiment (recall that many requests to Twitter are needed to process each *t.coURL* and the requests are rate-limited), we set the threshold slightly lower at 3 which increases recall but sacrifices on precision. For adversaries who has the resources to collect and process large number of *t.coURLs*, they could set this threshold higher to improve precision over recall.

5.1.3 Ranking Deduced Employees

After obtaining the list of deduced employees, we further rank them based on who they follow. The idea is that a user is more likely to be an employee if he or she follows a few other employees in the company. Therefore, for each deduced employee, we find out who he or she follows and how many of these followees appear in the list of deduced employees. There is no need to make any further request to Twitter because this information should have already been obtained when we retrieve the feeds of each derived follower in Section 5.1. If adversaries already know the Twitter accounts to some of the employees, they could use them in the comparisons.

Next, we rank (in decreasing order) the list of deduced employees according to the total number of followees who are also either a deduced employee or a previously known employee. Having a ranked list would improve efficiency when one needs to conduct further contextual reviews to confirm the deductions, especially if such reviews is done manually. For example, we could look for contexts relating to the company in their Twitter profiles or Tweets, or maybe link their accounts to possible Facebook or LinkedIn accounts which might provided more evidences of their employment.

A ranked list could also help those who want to further eliminate possible false positives in their deductions by removing those that do not follow anyone in both the list of deduced employees and previously known employees. This, however, should only be used when the list of deduced employees is going to be used with other techniques and the presences of false positives will skew outcome significantly. This is because the elimination could also possibly remove true positives since the list of deduced employees and previously known employees do not represent everyone in the company. Therefore, it should be used with discretion.

5.2 Verifying Anonymous PMDs

Our algorithm designed for verifying anonymous PMDs proceeds in three steps: First, even before any anonymous URL is observed, we can already send requests to Twitter to retrieve and cache the feeds received by each of the known employee using the two steps approach in Section 5.1. This two steps approach can optionally be made using the Streaming APIs instead of the REST APIs if real-time updates and cachings are required. Since the number of known employees are fixed and are usually not large, thus it can be achieved in a relatively cost-effective manner using either one of the APIs.

Next, after observing sufficiently large number of URLs from an anonymous PMD, we again extract all available *t.coURLs* from the observed URLs (as previously discussed in Section 3.3) and discard those that are not relevant (see Figure 5.7). Thirdly, we compare the extracted *t.coURLs* against those found in the feeds of each known employee using Equation 5.1. This time, however, we use in the same way as (Su et al., 2017) i.e. measure the likelihood that the *t.coURLs* in the observed URLs belong to those found in the feeds of each known employee. This last step is a straightforward comparison and do not require further requests to Twitter. Therefore, we can perform comparison for many anonymous PMDs without the need for huge computational resources and processing time. This is not possible if we follow the strategy discussed in (Su et al., 2017) where requests to Twitter are made after observing the URLs.

However, using Equation 5.1 alone only helps us to find the most likely employee that might be the owner of that anonymous PMD. It however, does not tell us how accurate is the suggested ownership. For example, even if there is only one *t.coURLs* that appears in the feeds of the most likely employee, that employee will still be deduced as the most likely owner of the anonymous PMD when none of the other *t.coURLs* appear in other employees' feeds. In this example, the most likely employee is definitely not a correct selection since majority of the *t.coURLs* did not appear in his or her feeds. To ensure that the selection is correct, we add Likelihood Ratio Testings (LRT) (Wikipedia, 2017b) on top of the MLE calculations to help us determine if the selected employee is more likely or more not likely to be the owner of the anonymous PMD.

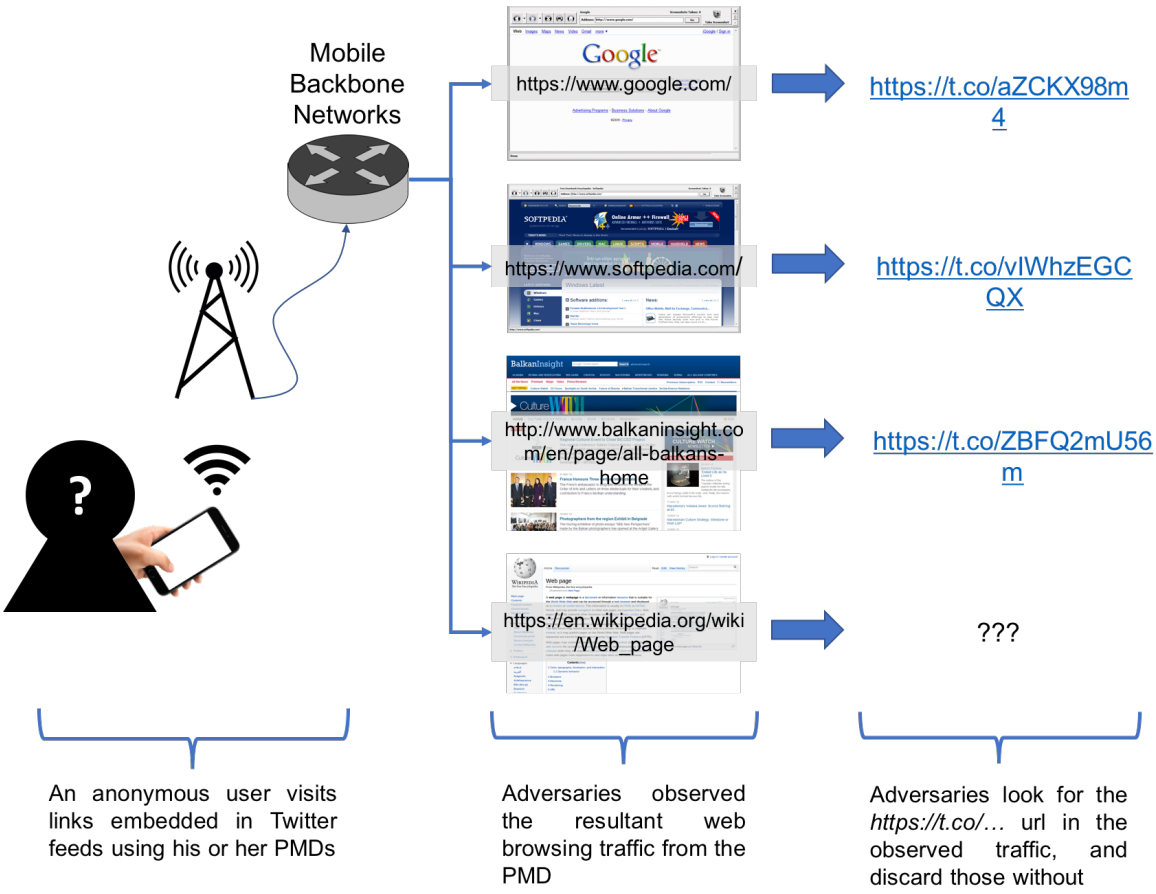


Figure 5.7: Extracting *t.coURLs* from URLs observed from an anonymous PMD.

The LRT is formulated as:

$$\begin{aligned} LR &= -2log[\frac{\widehat{U\ell}_{top}}{\widehat{\ell}_{top}}] \\ &= 2log[\frac{\widehat{\ell}_{top}}{\widehat{U\ell}_{top}}] \\ &= 2[\widehat{\ell}_{top} - \widehat{U\ell}_{top}] \end{aligned} \tag{5.2}$$

where $\widehat{\ell}_{top}$ denotes the likelihood that the observed *t.coURLs* appear in the feeds of the selected employee and is calculated using Equation 5.1. While $\widehat{U\ell}_{top}$ denotes the likelihood that the observed *t.coURLs* are not from the feeds of the selected employee and is calculated using the same Equation 5.1, but setting q_R as:

$$q_R = \frac{|U - L|}{|U|} \tag{5.3}$$

Our algorithm can be used to verify any kind of anonymous personal computing device such as laptop or desktop (provided that the user visits links in their Twitter feeds using the device), but we are particularly interested in mobile devices as there is a higher chance that employees will bring them to work. Web browsing traffic from mobile devices can be identified using browser fingerprinting techniques (Upathilake et al., 2015) and can be performed by both network eavesdroppers and malicious third-party sites. One possible method is to look for mobile OS signatures in the HTTP *user-agent* headers (e.g. "*Mozilla/5.0 (iPhone; CPU iPhone OS 10_0 like Mac OS X) AppleWebKit/602.1.38*").

Chapter 6

Evaluation

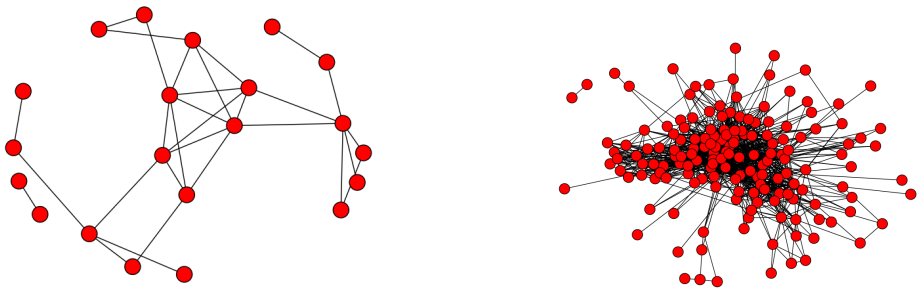
In this section, we evaluate the performance of our strategy using the Twitter social circles dataset from Stanford Large Network Dataset Collection (Leskovec and Krevl, 2014). We briefly review the background to this dataset, discuss how we adopt it in our experiments and explain how these experiments are conducted. We then analyse the experiments results and finally, examine if our strategy are exploitable by network eavesdroppers and malicious third-party sites.

6.1 Social Networks Dataset

As previously mentioned in Section 3.1, employees' social network structure in real-life companies could be anywhere between a simple single social circle, to one consisting of multiple interconnected social circles. Therefore, it is necessary to evaluate the performance of our strategy against many different possible social network structures. These evaluation results could also help security managers better understand how vulnerable their companies are to this kind of attack.

To test our strategy against a wide range of social network structures, we use the Twitter social circles dataset from the Stanford Large Network Dataset Collection (Leskovec and Krevl, 2014). This dataset enable us to experiment with social networks of known social circles with varying size and complexity, thereby simulating the employees' social network structures of different companies. We use clustering coefficient to measure the complexity of a social circle and calculate it using only reciprocal relationships. This is because clustering coefficient provides a good perspective on a graph's complexity as it measure the degree to which nodes in a graph tend to cluster together (Wikipedia, 2017a), and reciprocity suggest the possibility of offline friendships (Natali and Zhu, 2016; Kim et al., 2016) and thus are more reflective of the relationships between employees in a real-life company. Furthermore, reciprocity also affects *snowballing directed triadic closure* (see Section 3.1) and therefore, clustering coefficients calculated using reciprocal relationships better reflects how complexities of the networks affect the performance of our strategy.

There are in total 4065 social circles in the dataset with size ranging from 1 to 207 and have an average clustering coefficient of 0.471, which mirror those in small-and-medium enterprises. Figure 6.1 shows the social network structures of two Twitter social circles in the dataset. Due to the rate-limits imposed by Twitter APIs (see Section 3.3) and limited available computational resources, we are not able to evaluate our strategy against all the social circles in this dataset. Therefore, we randomly choose 5 of them with different sizes and clustering coefficients (see Table 6.1) for our experiments.



(a) Social network structure of size 21 and clustering coefficient 0.249.

(b) Social network structure of size 162 and clustering coefficient 0.546.

Figure 6.1: Examples of social network structures in Twitter social circles dataset from Stanford Large Network Dataset Collection.

No.	Size	Clustering Coefficient	Num. Connected Components
1	15	0.649	1
2	20	0.444	1
3	40	0.249	2
4	80	0.459	1
5	94	0.546	1

Table 6.1: Size and clustering coefficients of 5 Twitter social circles randomly chosen from the dataset.

6.2 Constructing Synthetic URLs

The performance of our strategy are evaluated against multiple sets of synthetic URLs. These sets of URLs are constructed with the assumption that users follow a simple behavioural model when visiting links in Twitter (Su et al., 2017): a user visits links mostly posted by their followees but will occasionally visit those that are posted by a followee of a followee. Links posted by followees-of-followees help to test our strategy's resilience to noise and it is reasonable to assume that a user is encouraged to visit such links either by Twitter's recommendation system (Most Innovative Company, 2014) or through his or her own exploration.

Based on this assumed behavioural model, we construct two sets of synthetic URLs for our experiments: The first set represents URLs that are observed from a company and are used for testing the algorithms for deducing employees' Twitter accounts, while the second set represents those observed from anonymous PMDs and are used for testing the algorithm for verifying if those PMDs belongs to one of the employees.

6.2.1 Company's URLs

The set of synthetic URLs observed from a company is constructed in three steps. First, for each of the 5 social circles, we select a set of reasonably active users to represent employees who visit links using their ODs. A reasonably active user is one who do not have too little or too many followees or followers. Users who have too little followees have lower chance of visiting links in their feeds due to the lack of feeds, while those who have too little followers (e.g. bots), or too many followees (e.g. spammers) or followers (e.g. celebrities) do not reflect the behaviour of a normal human user. Excluding users with too many followees and followers also helps to ensure that our experiments can be conducted using reasonable computational resources and completed within reasonable time. Therefore, we only select users who have between 20 to 300 followees and 20 to 1000 followers to construct the synthetic URLs.

Next, for each selected user, we send requests to Twitter to retrieve all links that have appeared in his or her feeds in the last 5 days via the two steps approach illustrated in Figure 5.4, and then randomly select N number of them for our experiments.

Finally, we generate and blend $\text{round}(0.2 * N)$ number of noise URLs for each selected user using a three steps approach: First, we sample a followee of the selected user uniformly at random, then we sample a followee of that followee at random, and finally sample a link posted by that followee-of-followee in the last 5 days. Table 6.2 shows a breakdown of the number of selected users and the total number of URLs constructed, for each of the 5 social circles. Note that the total number of URLs might not be equal to $(N + \text{round}(0.2 * N)) \cdot |\text{selected users}|$ because some of the selected users might either be private or do not have sufficient links in their feeds.

No.	Size	Num. Selected Users	Total Num. URLs $N + \text{round}(0.2 * N)$	
			12	**24**
1	15	5	60	-
2	20	13	100	-
3	40	20	145	255
4	80	34	392	700
5	94	50	445	809

Table 6.2: Breakdown of the number of users selected as active employees, and the total number of URLs constructed as URLs observed from the 5 social circles

6.2.2 Anonymous PMDs' URLs

The sets of URLs observed from anonymous PMDs are also constructed in three steps. We first select 4 sets of active anonymous users: 1) members belonging to each of the 5 social circles, 2) members' followees, 3) members' followers, and 4) other unrelated random users. We deliberately select users from the list of followers which were derived using the synthetic URLs constructed in Section 6.2.1 (following the process previously described in Section 5.1). This is because these users are more likely to share similar URLs than any other randomly chosen users since all of them receive at least of one the synthetic URLs in their feeds. This helps to better test the algorithm ability to differentiate non-employees who share URLs with the employees. We also separately test against the employees' followers, their followees, and other unrelated strangers so that we can better assess if the type of relationship affects the performance the algorithm.

From each set of selected users, we then randomly select M number of them for our experiments. We use the first set of users to test the accuracy of the algorithm in verifying PMDs that belong to the employees (i.e. True Positives), while we use the second, third and fourth sets to test the accuarcy of the algorithm in verifying PMDs that do not belong to the employees (i.e. True Negatives). We repeat steps 2 and 3 from Section 6.2.1 to construct the set of URLs for each of the selected user. Similarly, we deliberately reuse the URLs constructed in Section 6.2.1 as much as possible so that we can better test the algorithm ability to differentiate non-employees who share URLs with the employees. Table 6.3 shows a breakdown of the number of selected users in each of the 4 sets of anonymous users and the total number of URLs constructed, for each of the 5 social circles. Similarly the total number of URLs might not be equal to $(M + \text{round}(0.2 * M)) \cdot |\text{selected users}|$.

No.	Size	User Type	Num. Selected Users M	Total Num. URLs $N + \text{round}(0.2 * N)$					
				1	**6**	**12**	**24**	**36**	**60**
1	15	employee	5	5	30	60	120	169	281
		followee	50	50	287	574	1145	1718	2986
		follower	50	50	298	597	1195	1793	2984
		random	50	50	294	588	1050	1723	2987
2	20	employee	10	10	58	103	180	245	364
		followee	50	50	300	599	1193	1800	2995
		follower	50	50	300	597	1193	1792	2985
		random	50	50	300	599	1174	1793	2939
3	40	employee	17	17	85	145	217	267	306
		followee	50	50	300	599	1195	1792	2983
		follower	50	50	300	599	1197	1786	2983
		random	50	50	299	598	1196	1790	2923
4	80	employee	33	33	198	394	709	932	1197
		followee	50	50	300	600	1193	1792	2917
		follower	50	50	300	599	1196	1792	2982
		random	50	50	264	540	1030	1691	2815
5	94	employee	42	42	241	457	817	1092	1517
		followee	50	50	288	575	1103	1618	2634
		follower	50	50	288	576	1075	1611	2680
		random	50	50	228	466	935	1548	2515

Table 6.3: Breakdown of the number of selected users in each of the 4 sets of anonymous users and the total number of URLs constructed as URLs observed from their PMDs

6.3 Deducing Employees' Accounts

To evaluate the performance of the algorithms for deducing employees' Twitter accounts, we first mixed the URLs constructed in Section 6.2.1 together as a single test set and run it through the steps for 1) finding the most likely account, 2) finding other likely accounts, and 3) ranking the deduced accounts, as previously discussed in Sections 5.1, 5.1.1, 5.1.2 and 5.1.3. Since not all of the selected users will visit the same number of links, we repeat the tests over 100 times with different random number of URLs (i.e. between 1 and $N + \text{round}(0.2 * N)$) for each selected user in each run. The results for the different experiments are tabulated in Tables A.1, A.2 and A.3 respectively.

6.3.1 Top Likelihood Deduction

Table A.1 records the experiment results for the *Top Likelihood Deduction* algorithm which deduces the most likely Twitter account that could belong to one of the employees. Plotting the True Positive Rates (TPRs) of the algorithm's deduction against clustering coefficients (Figure 6.2) shows that deductions get more accurate with larger clustering coefficients, both for $N = 10$ and $N = 20$. Furthermore, we are able to deduce (with TPR > 0.925) at least one of the employees for social circles with clustering coefficients > 0.4. The plot also shows that accuracies increases slightly when more synthetic URLs are used for the testings.

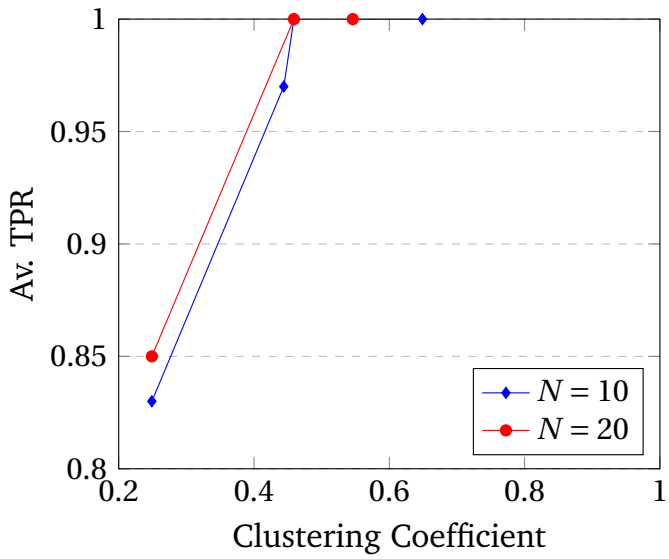
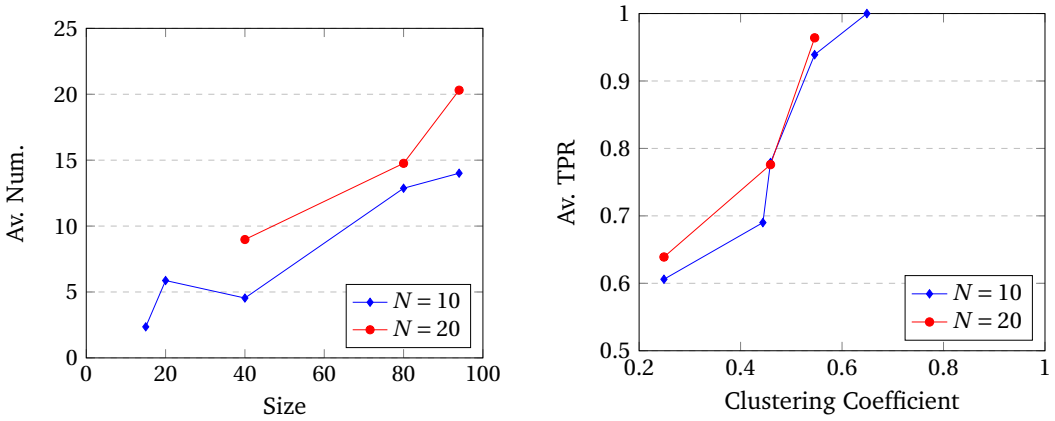


Figure 6.2: Plot of *Top Likelihood Deduction* algorithm's TPRs vs clustering coefficients

6.3.2 Eliminate & Iterate Likelihood Deduction

Next, Table A.2 records the experiment results for the *Eliminate & Iterate Likelihood Deduction* algorithm which deduces other likely Twitter accounts that might belong to the employees. The results show that more employees' Twitter accounts can be deduced using this algorithm, especially with more URLs (see Figure 6.3(a)). Plotting the TPRs of the algorithm's deduction against clustering coefficients (Figure 6.3(b)) again shows that deductions get more accurate with larger clustering coefficients, and achieve very high accuracy of TPR > 0.85 for social circles with clustering coefficients > 0.5.



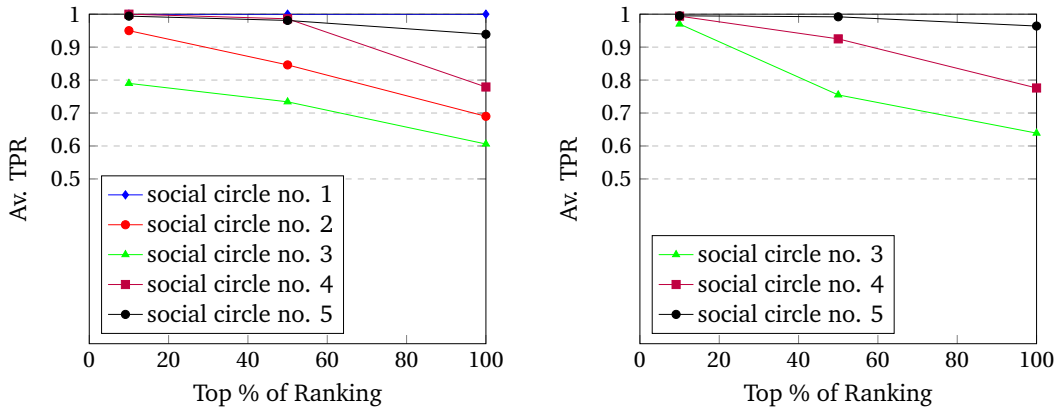
(a) Num. Deduced Employees vs size.

(b) TPR vs clustering coefficients.

Figure 6.3: Performance of *Eliminate & Iterate Likelihood Deduction* algorithm

6.3.3 Ranking Deductions

Finally, Table A.1 records the experiment results for ranking the Twitter accounts deduced using both the *Top Likelihood Deduction* and *Eliminate & Iterate Likelihood Deduction* algorithms. Plotting the deductions' TPRs against the top % of the ranking (Figure 6.4) shows that deduction accuracy increases as we move up the ranking. This proves that ranking the deduced accounts using the algorithm discussed in Section 5.1.3 helps to better prioritise the deductions.



(a) $N = 10$. (b) $N = 20$.

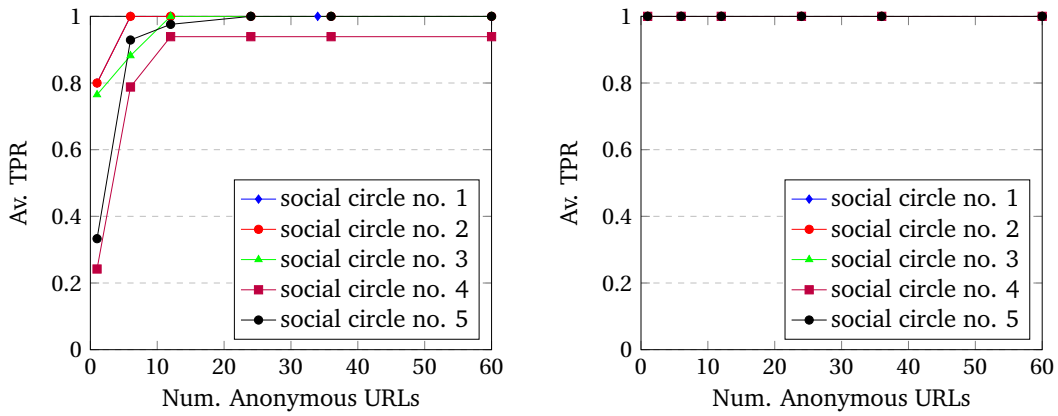
Figure 6.4: Plot of deduction TPR vs top % of ranking.

Based on the above three sets of experiments results, we could further predict that at least one of the users in most of the Twitter social circles from the Stanford Large Network Dataset Collection could be deduced confidently (i.e. with a relatively high accuracy of TPR > 0.8), since the average clustering coefficient of those social circles is around 0.471.

6.4 Verifying Anonymous PMDs

To evaluate the performance of the algorithm for verifying anonymous PMDs, we run the URLs constructed in Section 6.2.2 through the steps previously discussed in Section 5.2. The results for this experiment are tabulated in Tables A.4.

Figure 6.5 shows how the accuarcy for verifying anonymous PMDs belonging to employees varies with the number of anonymous URLs. The TPRs in Plot 6.5(a) measure the accuracy of verifying the exact identity of the employee that owns the anonymous PMD, while those in Plot 6.5(b) measure the accuracy of verifying that an anonymous PMD belong to one of the employee. The former shows that the smaller the size of the social circle, the lesser number of URLs are needed to differentiate between each employees, and at least 12 URLs are needed to achieve highly accurate verifications (i.e. TPR > 0.9). However, if we do not need to find the exact identity of the employee (i.e. enough to know that the anonymous PMD belongs to one of the employees), lesser URLs are needed to achieve accurate verifications (see Plot 6.5(b)).

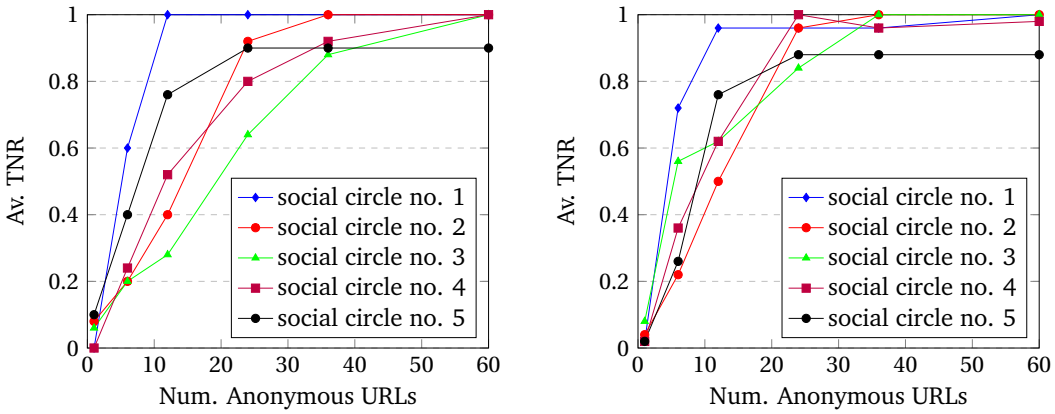


(a) Individual. (b) Group.

Figure 6.5: Plot of TPRs vs number of anonymous URLs, for verifying anonymous PMDs belonging to employees.

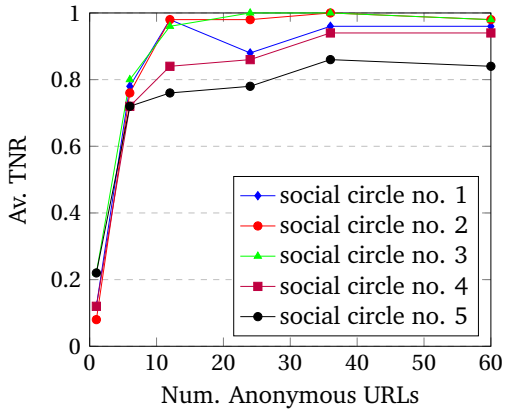
However, in real-life applications, the algorithm's True Negative Rate (TNR) performance is much more important than its TPR performance. This is because powerful adversaries observe URLs from millions of anonymous PMDs daily and most of them do not belong to the employees. As such, the algorithm will be used to verify non-employees' PMDs most of the time, and having poor TNR performances will result in a lot of non-employees' PMDs being accepted frequently.

Figure 6.6 shows the algorithm's TNR performances for verifying anonymous PMDs belonging to a) employees' followees, b) employees' followers, and c) users who have no relationships with any of the employees (i.e. strangers). Results show that verification accuracy increases (or the number of URLs needed to achieve highly accurate verification reduces) as the relationship between the owner of the anonymous PMD and the employees weakens (i.e. from Followee to Follower to Stranger). This is especially true for social circles with low clustering coefficients. The need for more URLs to verify employees's followees and followers as compared to strangers proves that *directed triadic closure* leads to the sharing of more feeds between two users, thus making it more difficult to differentiate between them. Since in real-life applications we do not know what type of relationship the owner of an anonymous PMD has with the employees, Plot 6.6(a) can be taken as the worst case scenario, which shows that at least 36 URLs are needed to achieve highly accurate verifications (i.e. TNR > 0.9).



(a) Employees' Followees.

(b) Employees' Followers.



(c) Strangers.

Figure 6.6: Plot of TNRs vs number of anonymous URLs, for verifying anonymous PMDs that do not belong to employees.

6.5 Adversary Limitations

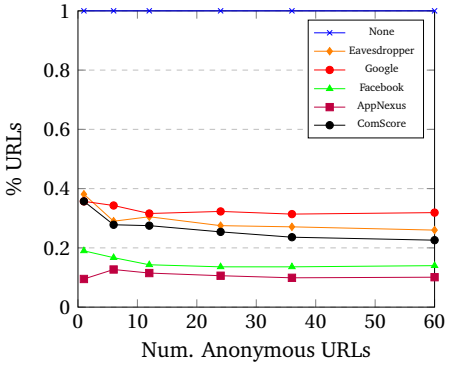
As previously discussed in Chapter 4, adversaries are not able to observe of all the URLs that are generated: Network eavesdroppers are not able to see those that are encrypted, while malicious third-party sites are not able to see those that belong to first-party websites which they have no coverage. Such limitations reduce the number of URLs that adversaries could exploit for both 1) deducing the employees' Twitter accounts and 2) verifying anonymous PMDs. However, the former could be easily overcomed by extending the monitoring duration for a very long period until sufficient URLs needed to yield relatively accurate results are collected. Extending the monitoring is achievable because companies usually lease fixed public IP addresses from their ISPs which make them easily trackable over the Internet. On the contrary, public IP addresses of PMDs change frequently, making it impossible to track them persistently and thus gives adversaries much shorter window of opportunities for collection. As such, we foresee that adversaries will only be affected by the lack of full URLs observation when trying to verify anonymous PMDs.

Therefore, to study how adversaries' limitations impact our strategy, we re-evaluate the algorithm for verifying anonymous PMDs using reduced sets of the synthetic URLs constructed in Section 6.2.2: 1) To simulate the limitations faced by network eavesdroppers, we remove URLs that are encrypted (i.e. those that use the HTTPS protocol); 2) To simulate the limitations faced by malicious third-party sites, we remove URLs that are not tracked by the four major third-party trackers (i.e. Google, Facebook, AppNexus and Comscore). To help us determine which third-party trackers are tracking the URLs, we make use of OpenWPM (Englehardt and Narayanan, 2016) to automate web browsing interactions with the URLs and record the third-party requests. Finally, we repeat the experiment in Section 6.4 using these five sets of reduced URLs and tabulate the results in Table A.4.

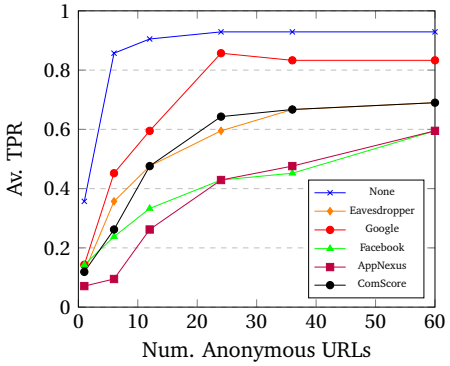
Both the TPRs plots in Figures 6.7(b) and 6.7(c), and the TNRs plots in Figures 6.8(b), 6.9(b) and 6.10(b) show that the performance of the algorithm drops when URLs are removed. And when we compare the drop in performances against the corresponding reduction amount in Figures 6.7(a), 6.8(a), 6.9(a) and 6.10(a), we can see that the more URLs that are removed, the poorer the performances. This is the reason why, unlike the trends displayed in Section 6.4, verification accuracy weakens as the relationships between the owner of the anonymous PMD and the employees weakens.

The results from these experiments illustrate the restrictions adversaries faced when applying our algorithm in real-life. They are only able to accurately verify PMDs belonging to users who actively visit links in their Twitter feeds before their public IP address changes. Taking reference from the worst case scenario in Figure 6.10, this will translate to having an accuracy of TNR > 0.7 for users who visit more than 60 links. From the perspective of the adversaries, this means that they will need to collect a minimum of $0.2 \cdot 60 = 10$ URLs (see Figure 6.10(a)) from each anonymous

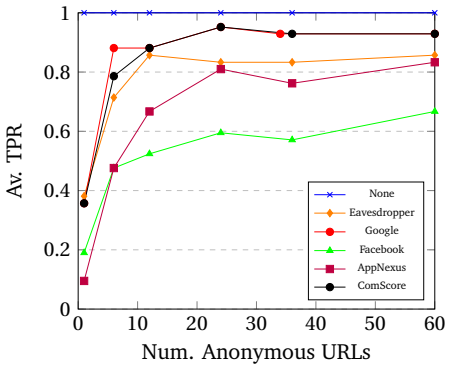
PMD, in order to achieve similar accuracy. This is still feasible in real-life applications based on a large-scale study done by Casado and Freedman (2007), which shows that fewer than 2% of clients uses more than two IP addresses in a week and only 8% of clients uses more than three in a month.



(a) % URLs left after reduction.

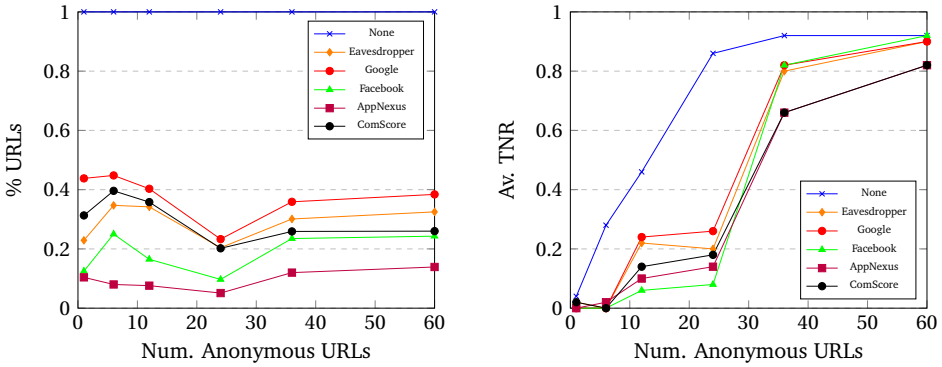


(b) Av. TPRs (Individual).



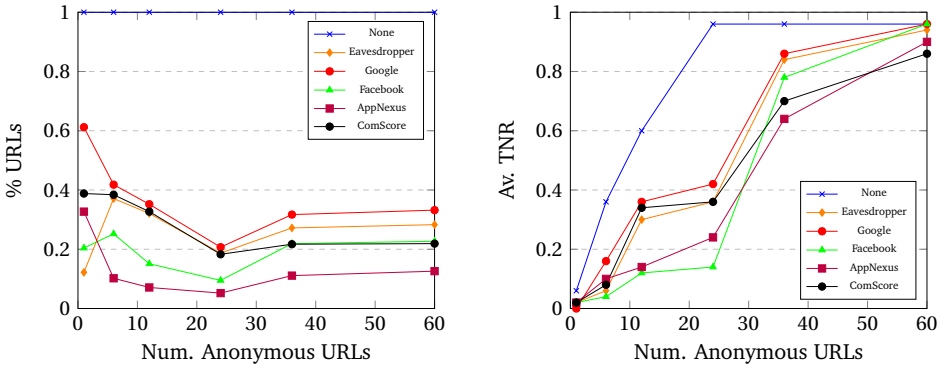
(c) Av. TPRs (Group).

Figure 6.7: Plot of % URLs and TPRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs that belonging to employees.



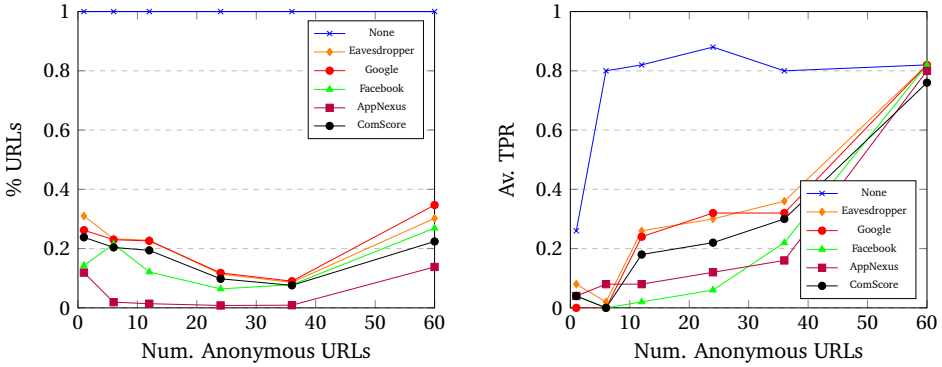
(a) % URLs left after reduction. (b) Av. TNRs.

Figure 6.8: Plot of % URLs and TNRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs belonging to employees' followees.



(a) % URLs left after reduction. (b) Av. TNRs.

Figure 6.9: Plot of % URLs and TNRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs belonging to employees' followers.



(a) % URLs left after reduction. (b) Av. TNRs.

Figure 6.10: Plot of % URLs and TNRs vs number of anonymous URLs (for different adversaries), for verifying anonymous PMDs belonging to strangers.

Chapter 7

Countermeasures

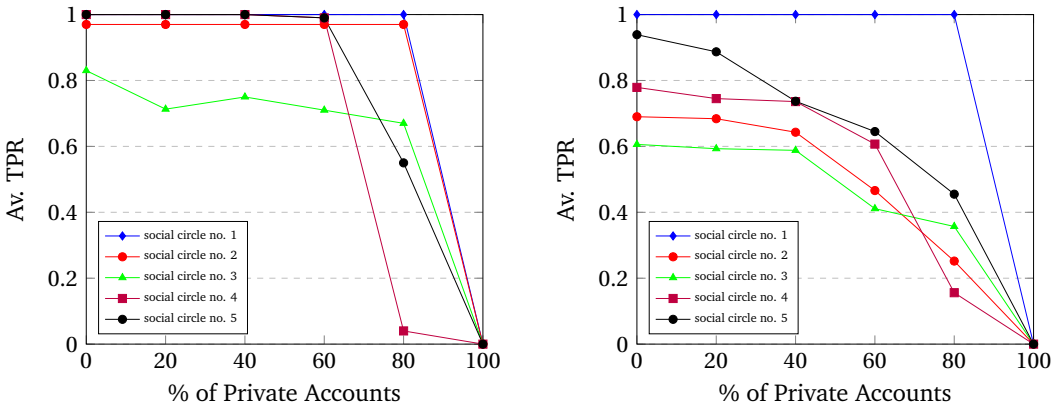
In this section, we discuss the different measures that could help security managers to counter our strategy and thereby preventing adversaries from finding employees' PMDs when they are outside the protection of the company network firewalls. In turn, this will prevent their devices from being exploited as trojan horses to launch indirect attacks on the company and its networks.

7.1 Switching Accounts to Private

The most straightforward method is to advise employees to protect their Twitter accounts by switching them to private. Twitter accounts are public by default but employees can restrict public access by following the instructions given in (Twitter, 2017d). When an account has been made private, adversaries will not be able to retrieve its list of followers, and consequently its feeds from Twitter (see Figure 5.4). This is true even if they are not using Twitter APIs but crawl the Twitter website to scrape for the required information, as only existing followers are able to view who a private user is following from his or her Twitter profile page (Nield, 2017). If they attempt to follow the private user on Twitter to gain access to the required information, they risk potential exposure because now the user will need to approve the followership. (Twitter, 2017c).

Consequently, without the feeds, adversaries will not be able to calculate the employees MLE scores and thus making our strategy useless. This works both when adversaries are trying to deduce employees' Twitter accounts using URLs observed from the company, as well as when they are verifying anonymous PMDs. To evaluate how the switching of employees' Twitter accounts private will impact the performance of our strategy, we repeat the experiments for 1) finding the most likely account and 2) finding other likely accounts (as previously discussed in Section 6.3) with different number of employees' accounts being switched to private (which we simulate by removing their feeds when we calculate their MLE scores). The results of the experiments are tabulated in Tables A.7 and A.8 respectively.

Plotting the TPRs against the % of private employees' accounts shows that as more employees switches their accounts to private, the less likely adversaries are able to deduce them from URLs observed from the company (see Figure 7.1). This proves that switching employees' account to private is effective in protecting the employees from adversaries who exploit our strategy.



(a) *Top Likelihood Deduction* algorithm.

(b) *Eliminate & Iterate Likelihood Deduction* algorithm.

Figure 7.1: Plot of TPRs vs % of private Twitter Accounts, for each social circle

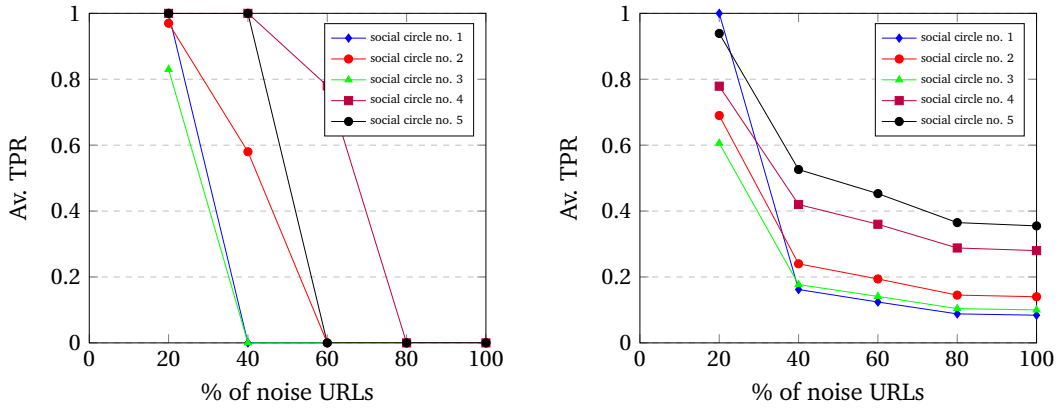
7.2 Inducing Noise URLs

Although advising employees to switch their Twitter accounts to private is an effective countermeasure, experiments results indicate that nearly all the employees need to switch their account to private in order to achieve a relatively low deduction accuracy of TPR < 0.2. Some employees might be reluctant to do so as they might interpret this as an attempt to interfere with their personal lives, or fearing the repercussions (e.g. attracting lesser followers) of doing so (Twitter, 2017c; Aase, 2010). Therefore, alternatives are needed for companies who could not convince enough employees to make the switch.

One alternative countermeasure is to deliberately induce more noise to the observed URLs. Previously in Section 6.2, we mentioned that employees, on top of links that appear in their feeds, might visit other links that are either recommended by Twitter or through their own exploration. The URLs generated from these links are noise URLs and security managers could further induce more of them by deliberately searching and visiting such links. For example, security managers could use Twitter APIs to automatically search for links that are not posted by any of the employees' followees, and use OpenWPM to automate the visiting of those links. Since the generated URLs will be mixed together by the company's NAT gateway, adversaries will never know that the observed URLs contain deliberately induced noises.

The previous experiments results have shown that our strategy is resilience to 20% noise ratio. To evaluate how increasing the number of noise URLs will reduce the performance of our strategy, we repeat the experiments for 1) finding the most likely account and 2) finding other likely accounts (as previously discussed in Section 6.3) again with different amount of noise URLs. The results of the experiments are tabulated in Tables A.9 and A.10 respectively.

Plotting the TPRs against the % of noise URLs shows that as more noises are induced, the less likely adversaries are able to deduce employees from URLs observed from the company (see Figure 7.2). This proves that inducing noise URLs is another effective method for protecting the employees from adversaries who exploit our strategy.



(a) *Top Likelihood Deduction* algorithm.

(b) *Eliminate & Iterate Likelihood Deduction* algorithm.

Figure 7.2: Plot of TPRs vs % of noise URLs, for each social circle

7.3 Decoy Countermeasures

Although advising employees to switch their accounts to private and inducing additional noise URLs should already be very effective in countering our strategy, they will result in more innocents Twitter users (i.e. non-employees) falling victims to the adversaries exploitations. As we believe that a socially responsible countermeasure should not cause harm to innocents victims, we further explore two other methods that will mislead the adversaries to decoys instead of existing Twitter accounts.

7.3.1 Decoys Mirroring Employees' Followerships

We can design a decoy countermeasure by creating decoy Twitter accounts that mirror the followerships of the employees. As previously discussed in Section 5.1.1, it is not necessary for the deduced users to physically visit any links from their feeds: As long as majority of the relevant links appearing in his or her feeds, they are likely to be deduced by both the *Top Likelihood Deduction* and *Eliminate & Iterate Likelihood*

Deduction algorithms. This also means that the deduced user is most probably someone that follows the most number of users that have posted those links.

Therefore theoretically, if we redistribute all the employees' followees equally among a much smaller group of decoys as illustrated in Figure 7.3, each decoy will receive a significant portion of the links that would potentially generate the URLs observed by the adversaries, thereby increases their chances of being deduced by the two algorithms. To evaluate how deploying decoys impact the performance of our strategy, we repeat the experiments in Section 6.3 for 1) finding the most likely account and 2) finding other likely accounts again, but this time with different number of decoy accounts. The results of the experiments are tabulated in Tables A.11 and A.12 respectively.

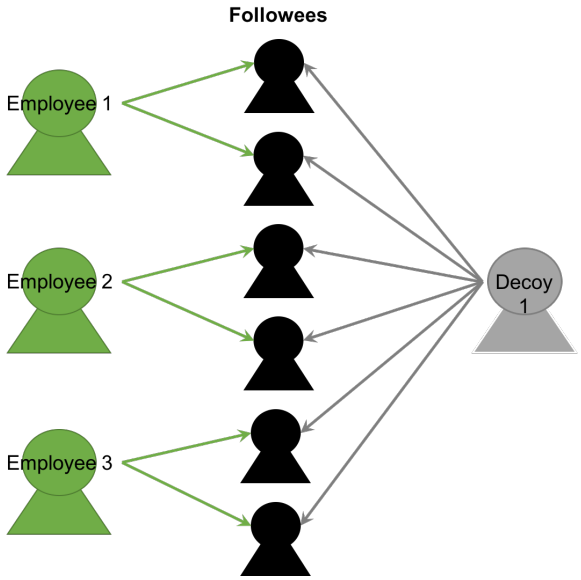
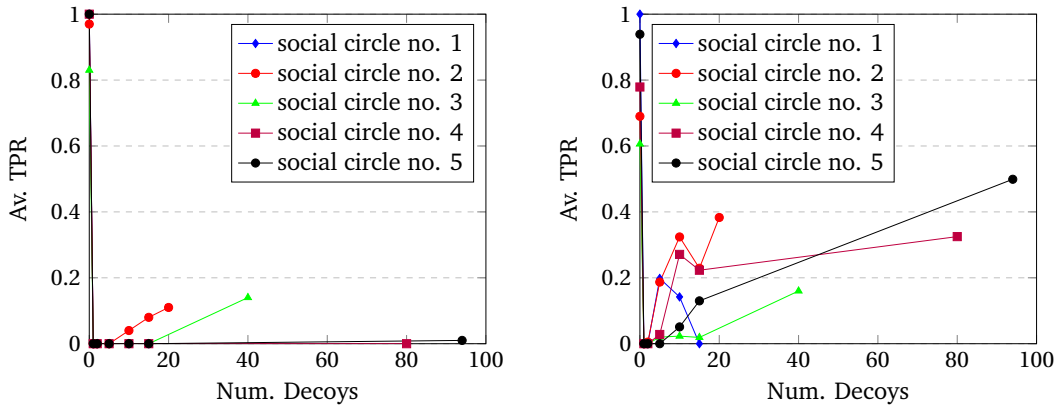


Figure 7.3: Redistributing employees' followees among a smaller group of decoys.

Plotting the TPRs against the number of decoys shows that having one or two decoys that follow all the employees' followees yield the best performance for this countermeasure, and performance generally degrades with more decoys (see Figure 7.3). This trend is reasonable since having more decoys means that each decoy follow less of the employees' followees, and thus decreases the opportunity for them to receive the majority of the links that generate the observed URLs. As such, having a small number of decoys generally would be effective in protecting the employees from adversaries who exploit our strategy.



(a) *Top Likelihood Deduction* algorithm

(b) *Eliminate & Iterate Likelihood Deduction* algorithm

Figure 7.4: Plot of TPRs vs number of decoys, for each social circle.

7.3.2 Decoys Following Radom Users

We can also design a decoy countermeasure by creating decoy Twitter accounts that follows random users who are not related to the employees. The idea is for the decoys to physically visit many links in their feeds so that significant noise URLs exist in the URLs observed by the adversaries, and thereby degrades the performances of the *Top Likelihood Deduction* and *Eliminate & Iterate Likelihood Deduction* algorithms. This is similar to the method of inducing random noise URLs as discussed in Section 7.2, but instead of using Twitter APIs to search for random links that are not posted by the employees' followees, we search for random users that are not related to the employees and have our decoys follow them (as illustrated in Figure 7.5). We can also use OpenWPM to automate the visiting of links posted by those random users. Since the links that generate the noise URLs now appear in the decoys' feeds, the decoys (instead of the employees or any other innocent users) are thus more likely to be deduced by our algorithms as more noise URLs are generated. This is proven by the results of the experiments conducted in Section 7.2.

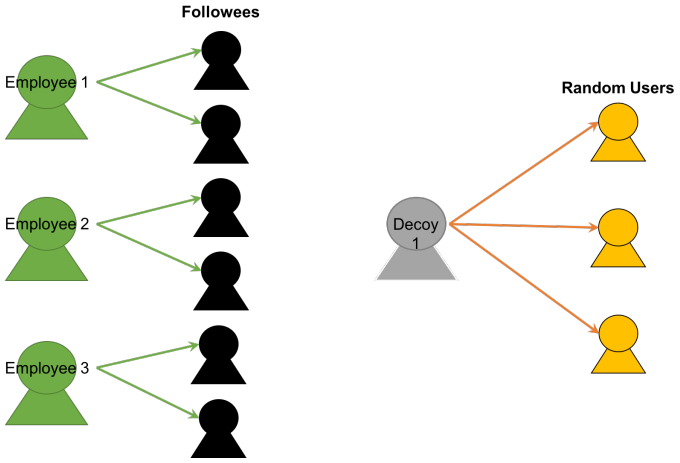


Figure 7.5: Creating decoys that follow other random users who are not employees' followees.

7.3.3 Avoiding Decoys Detection

Although creating decoys do serve as good and socially responsible countermeasures, it alone is not practical in real-life applications. This is because Twitter detects such accounts as spammers that practice aggressive following and will either lock or limit them (Twitter, 2017b). Likewise, adversaries could also detect and reject them from their deduction process easily. Therefore, we need to make the decoys more 'human', and this could be achieved by having more decoys so that we can reduce the number of followees per decoy, as well as encouraging followees to reciprocate the relationship. The former is easily achievable and aims to reduce the number of followees so that it falls within the range of a reasonably active human user (e.g. 100 to 500), and the latter aims to mimic offline friendships, and can be achieved using the technique proposed in (Digiarty Software, Inc., 2010) i.e. running a contest that requires participants reciprocate the relationships.

7.4 Alternative Countermeasures

The above four suggested methods are practical countermeasures that can be easily deployed in real-life applications. Nevertheless, there are also methods that could work in theory but are hard to deploy in real-life applications. Here, we review two such methods to wrap up our discussion on possible measures to counter our strategy.

First, it is possible to degrade the performance of our deduction algorithms by growing the social circle with a lot of non-employees. When the social circle includes a lot of non-employees that are also sharing similar followees with the employees, it reduces the likelihood of existing employees being deduced. This is especially true when the non-employees themselves already belong to another densely clustered so-

cial circle. However, growing the social circle takes time and cannot be controlled by the security managers: They first have to encourage both their employees and non-employees to follow each other and after that, let the *snowballing directed triadic closure* process run its course.

Another theoretical countermeasure is for the security managers to actively monitor and block URLs from egressing the company NAT gateway. As previously discussed in Section 3.3.1, URLs can be easily detected by looking out for *t.coURLs* in the web browsing traffic. Security managers also do not need to block all of the URLs, just those that are either posted by a small number users or those that are posted by users who has a small number of followers. This is because, using the methods described in Section 5.1, such URLs will derive a smaller sets of candidates for the deductions and thereby reducing the number of false positives. However, although blocking the company's web browsing traffic is within the rights of the company, it is very disruptive to the employees' browsing experience and thus defeats our origin objective of not having drastic measures that will bring about significant limitations to the employees. Furthermore, this method is very hard to implement as the network intrusion detection system needs to check with Twitter (i.e. using Twitter APIs) on the eligibility of each URL in real-time. Given the inherent latency and rate-limits of Twitter RESTful APIs, this might not even be possible in real-life applications. A more practical alternative is to block all of the URLs, but it simply just brings more disruption to the employees' browsing experience and is thus not encouraged.

Chapter 8

Conclusion and Recommendations

In this report, we demonstrated that it is possible for adversaries such as network eavesdroppers and malicious third party sites to accurately track the employees of a company across their office desktops (ODs) and personal mobile devices (PMDs), using social media networks. We devised a cross-device tracking strategy that leverages on links found in social media feeds to achieve this objective, and designed algorithms that accurately 1) deduce employees' social media accounts using web browsing traffic observed from the company, as well as 2) verify anonymous PMDs using web browsing traffic observed from those PMDs.

We tested both sets of algorithms on a popular social media service (Twitter): For the algorithms that deduce employees' social media accounts, experiments results show that one can achieve highly accurate deductions (i.e. TPR $>$ 0.8) if one is concerned only with the identities of a few employees. Accuracy also gets better when employees' social network structure becomes more complex (i.e. having larger clustering coefficients); For the algorithm that verifies anonymous PMDs, experiments results show that one can achieve highly accurate verifications (i.e. TPR and TNR $>$ 0.8) if one is able to observe more than 25 URLs from the anonymous PMD. However, accuracy is expected to be affected by the limitations (such as encryption and coverage) that adversaries face in real-life deployments. Nevertheless, adversaries could skip this phase of the tracking if they are unsuccessful, and embark on spear-phishing attacks to exploit employees' PMDs using the deduced Twitter accounts (Seymour and Tully, 2016).

We also demonstrated that it is possible to counter our strategy by preventing employees' Twitter accounts from being deduced accurately. This is achieved either by advising the employees to switch their accounts to private, or artificially induce more noises into the web browsing traffic observed from the company. We also explored alternative countermeasures that are more socially responsible (i.e. do not lead adversaries to innocent users). This is achieved by creating decoy Twitter accounts that either follow who the employees follow, or follow any other random users who are not related to the employees.

Finally, it must be noted that we conducted our experiments on synthetically crafted

web browsing traffic and use social networks that might not belong to actual employees of a company. On top of that, we only used five different sets of social networks which do not comprehensively represent those seen in real-life companies. Therefore, we recommend to further explore the feasibility of our strategy using more sets of social networks found in the Twitter social circles dataset from the Stanford Large Network Dataset Collection, as well as testing it with datasets collected from real-life companies and PMDs (including those owned by the employees). The former also helps to compile a set of benchmarks that security managers could reference when they want to determine if their employees' are vulnerable to cross-device tracking attacks by adversaries who exploit our strategy.

Bibliography

L. Aase. Twitter 135: 10 Reasons to NOT Protect Your Tweets, 2010. URL http://social-media-university-global.org/2010/06/twitter-135-10-reasons-to-not-protect-your-tweets/. pages 63

T. Anand and O. Renov. Machine learning approach to identify users across their digital devices. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1676–1680. IEEE, 2015. pages 9

M. Atkinson. 6 Reasons Employers Should Allow Smartphones in the Workplace, 2016. URL http://www.agencycentral.co.uk/articles/2016-09/allowing-smartphones-in-the-workplace.htm. pages 2

C. Banse, D. Herrmann, and H. Federrath. Tracking users on the internet with behavioral patterns: Evaluation of its practical feasibility. In *IFIP International Information Security Conference*, pages 235–248. Springer Berlin Heidelberg, 2012. pages 12

M. Bashir, S. Arshad, W. Robertson, and C. Wilson. Tracing information flows between ad exchanges using retargeted ads. In *Proceedings of the 25th USENIX Security Symposium In Proceedings of the 25th USENIX Security Symposium*, 2016. pages 17

R. Bilton. Cross-device tracking, explained, 2015. URL https://digiday.com/media/deterministic-vs-probabilistic-cross-device-tracking-explained-normals/. pages 9

J. Brookman, P. Rouge, A. Alva, and C. Yeung. Cross-Device Tracking: Measurement and Disclosures. In *Proceedings on Privacy Enhancing Technologies*, number 2, pages 133–148, 2017. pages 3, 8

X. Cao, W. Huang, and Y. Yu. Recovering Cross-Device Connections via Mining IP Footprints with Ensemble Learning. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1681–1686. IEEE, 2015. pages 10

M. Casado and M. Freedman. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *Proceedings of the 4th USENIX conference on Networked systems design & implementation 2007 Apr 11*, pages 13–13. USENIX Association, 2007. pages 37, 60

A. Cecaj, M. Mamei, and F. Zambonelli. Re-identification and information fusion between anonymized CDR and social network data. In *Journal of Ambient Intelligence and Humanized Computing*, pages 83–96, 2016. pages 16

CIKM. CIKM Cup 2016, 2016. URL `http://cikm2016.cs.iupui.edu/cikm-cup/`. pages 9

M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th international conference on World wide web*, pages 281–290. ACM, 2010. pages 38

R. Díaz-Morales. Cross-Device Tracking: Matching Devices and Cookies. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1699–1704. IEEE, 2015. pages 9

Digiarty Software, Inc. A Smart Way to Encourage People to Follow You Back on Twitter, 2010. URL `https://www.winxdvd.com/blog/encourage-people-to-follow-you-back-on-twitter.htm`. pages 67

Drawbridge. ICDM Contest Workshop on Cross-Device Connections, 2015. URL `https://drawbridge.com/blog/p/icdm-contest-workshop-on-cross-device-connections`. pages 9

D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press., 2010. pages 19, 21, 23

S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of ACM CCS 2016*, 2016. pages 59

S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, pages 289–299. ACM, 2015. pages 9, 35

Federal Trade Commission. Cross-Device Tracking, An FTC Workshop, 2015. URL `https://www.ftc.gov/news-events/events-calendar/2015/11/cross-device-tracking`. pages 7

Federal Trade Commission. FTC Issues Warning Letters to App Developers Using 'Silverpush' Code, 2016. URL `https://www.ftc.gov/news-events/press-releases/2016/03/ftc-issues-warning-letters-app-developers-using-silverpush-code`. pages 7

R. Gallagher and G. Greenwald. How the NSA Plans to Infect 'Millions' of Computers with Malware, 2014. URL `https://theintercept.com/2014/03/12/nsa-plans-infect-millions-computers-malware/`. pages 3

R. Gallagher and P. Maass. Inside the nsa's secret efforts to hunt and hack system administrators, 2014. URL `https://theintercept.com/2014/03/20/inside-nsa-secret-efforts-hunt-hack-system-administrators/`. pages 2

N. Giandomenico. WHAT IS SPEAR-PHISHING? DEFINING AND DIFFERENTIATING SPEAR-PHISHING FROM PHISHING, 2017. URL `https://digitalguardian.com/blog/what-is-spear-phishing-defining-and-differentiating-spear-phishing-and-phishing`. pages 3

M. Granovetter. The strength of weak ties. *American journal of sociology*, 78(6): 1360–1380, 1973. pages 18, 21, 23

X. Gu, M. Yang, C. Shi, Z. Ling, and J. Luo. A novel attack to track users based on the behavior patterns. *Concurrency and Computation: Practice and Experience*, 2016. pages 12

D. Herrmann, C. Banse, and H. Federrath. Behavior-based tracking: Exploiting characteristic patterns in DNS traffic. In *Computers and Security*, volume 39, pages 17–33, 2013. pages 12

C. Jackson, A. Bortz, D. Boneh, and J. Mitchell. Protecting browser state from web privacy attacks. In *Proceedings of the 15th international conference on World Wide Web*, pages 737–744. ACM, 2006. pages 14

M. Jakobsson and S. Stamm. Invasive browser sniffing and countermeasures. In *Proceedings of the 15th international conference on World Wide Web*, pages 523–532. ACM, 2006. pages 14

S. Kane, A. Karlson, B. Meyers, P. Johns, A. Jacobs, and G. Smith. Exploring cross-device web use on PCs and mobile devices. *Human-Computer Interaction–INTERACT 2009*, pages 722–735, 2009. pages 12

G. Kejela and C. Rong. Cross-device consumer identification. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1687–1689. IEEE, 2015. pages 9

M. Kim, J. Liu, X. Wang, and W. Yang. Connecting Devices to Cookies via Filtering, Feature Engineering, and Boosting. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1690–1694. IEEE, 2015. pages 9

Y. Kim, K. Choi, and F. Natali. Extending the Network: the Influence of Offline Friendship to Twitter Network, 2016. pages 20, 49

M. Kirchler, D. Herrmann, J. Lindemann, and M. Kloft. Tracked Without a Trace: Linking Sessions of Users by Unsupervised Learning of Patterns in Their DNS Traffic. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pages 23–34. ACM, 2016. pages 12

M. Landry and R. Chong. Multi-layer classification: Icdm 2015 drawbridge cross-device connections competition. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1695–1698. IEEE, 2015. pages 10

Lennarthaagsma. Deep dive into QUANTUM INSERT, 2015. URL `https://blog.fox-it.com/2015/04/20/deep-dive-into-quantum-insert/`. pages 37

J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection, 2014. URL `http://snap.stanford.edu/data`. pages 49

J. Lian and X. Xie. Cross-Device User Matching Based on Massive Browse Logs: The Runner-Up Solution for the 2016 CIKM Cup. arXiv preprint arXiv:1610.03928, 2016. pages 10

I. Lunden. 6.1B Smartphone Users Globally By 2020, Overtaking Basic Fixed Phone Subscriptions, 2015. URL `https://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phone-subscriptions/`. pages 3

C. Ma, D. Yau, N. Yip, and N. Rao. Privacy vulnerability of published anonymous mobility traces. In *IEEE/ACM Transactions on Networking (TON)*, pages 720–733, 2013. pages 15, 16

B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson. An analysis of china's "great cannon". In *FOCI. USENIX*, 2015. pages 37

Most Innovative Company. Don't Like Twitter's Recommended Tweets? Too Bad, They're Here To Stay, 2014. URL `https://www.fastcompany.com/3037277/dont-like-twitters-recommended-tweets-too-bad-theyre-here-to-stay`. pages 51

F. Naini, J. Unnikrishnan, P. Thiran, and M. Vetterli. Where you are is who you are: User identification by matching statistics. In *IEEE Transactions on Information Forensics and Security*, pages 358–372, 2016. pages 15

A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium*, pages 173–187. IEEE, 2009. pages 13, 15

F. Natali and F. Zhu. A Comparison of Fundamental Network Formation Principles Between Offline and Online Friends on Twitter. In *International Conference and School on Network Science*, pages 169–177. Springer, Cham, 2016. pages 20, 49

D. Nield. How to Hide Your Followers & Who You Are Following on Twitter, 2017. URL `http://classroom.synonym.com/hide-followers-following-twitter-13288.html`. pages 62

H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 597–606. ACM, 2007. pages 11

L. Olejnik, C. Castelluccia, and A. Janc. On the uniqueness of web browsing history patterns. In *annals of telecommunications-annales des télécommunications*, pages 63–74, 2014. pages 11

T. Patange. How to defend yourself against MITM or Man-in-the-middle attack, 2013. URL `https://hackerspace.kinja.com/how-to-defend-yourself-against-mitm-or-man-in-the-middl-1461796382`. pages 3

A. Rapoport. Spread of information through a population with socio-structural bias: I. Assumption of transitivity. *Bulletin of Mathematical Biology*, 15(4):523–533, 1953. pages 18

C. Riederer, Y. Kim, A. Chaintreau, N. Korula, and S. Lattanzi. Linking Users Across Domains with Location Data: Theory and Validation. In *Proceedings of the 25th International Conference on World Wide Web*, pages 707–719. International World Wide Web Conferences Steering Committee, 2016. pages 15

D. Romero and J. Kleinberg. The directed closure process in hybrid social-information networks, with an analysis of link formation on Twitter. In *ICWSM*, 2010. pages 4, 18

L. Rossi and M. Musolesi. It's the way you check-in: identifying users in location-based social networks. In *Proceedings of the second ACM conference on Online social networks*, pages 215–226. ACM, 2014. pages 16

L. Rossi, J. Walker, and M. Musolesi. Spatio-temporal techniques for user identification by means of GPS mobility data. In *EPJ Data Science*, page 11, 2015a. pages 15

L. Rossi, M. Williams, C. Stich, and M. Musolesi. Privacy and the city: User identification and location semantics in location-based social networks. arXiv preprint arXiv:1503.06499, 2015b. pages 16

L. Selsaas, B. Agrawal, C. Rong, and T. Wiktorski. AFFM: Auto feature engineering in field-aware factorization machines for predictive analytics. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1705–1709. IEEE, 2015. pages 9

J. Seymour and P. Tully. Weaponizing data science for social engineering: Automated E2E spear phishing on Twitter. In *Black Hat USA*, 2016. pages 37, 38, 69

R. Song, S. Chen, B. Deng, and L. Li. eXtreme gradient boosting for identifying individual users across different digital devices. In *International Conference on Web-Age Information Management*, pages 43–54. Springer International Publishing, 2016. pages 9

J. Su, A. Shukla, S. Goel, and A. Narayanan. De-anonymizing web browsing data with social networks. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1261–1269. International World Wide Web Conferences Steering Committee, 2017. pages 1, 3, 4, 5, 13, 14, 22, 25, 26, 28, 29, 31, 39, 42, 46, 51

L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. In *International Journal of Human-Computer Studies*, pages 97–137, 1997. pages 11

Y. Tay, C. Phan, and T. Pham. Cross Device Matching for Online Advertising with Neural Feature Ensembles: First Place Solution at CIKM Cup 2016. arXiv preprint arXiv:1610.07119, 2015. pages 10

Technical Center of Internet. DNS protocol doesn't provide data protection against eavesdropping, 2016. URL https://www.tcinet.ru/en/press-centre/news/4163/. pages 12

L. Tonsager. Digital Advertising Alliance Will Begin Enforcing its Cross-Device Guidance February 1, 2017, 2016. URL https://www.insideprivacy.com/advertising-marketing/digital-advertising-alliance-will-begin-enforcing-its-cross-device-guidance-february-1-2017/. pages 7

N. Tran. Classification and Learning-to-rank Approaches for Cross-Device Matching at CIKM Cup 2016. arXiv preprint arXiv:1612.07117, 2016. pages 10

Twitter. Twitter Developer Documentation, 2017a. URL https://dev.twitter.com/rest/reference. pages 5

Twitter. Following rules and best practices, 2017b. URL https://support.twitter.com/articles/68916. pages 67

Twitter. About public and protected Tweets, 2017c. URL https://support.twitter.com/articles/14016. pages 62, 63

Twitter. Protecting and unprotecting your Tweets, 2017d. URL https://support.twitter.com/articles/20169886. pages 62

Twitter. About Twitter's link service (http://t.co), 2017e. URL https://support.twitter.com/articles/109623. pages 31

R. Upathilake, Y. Li, and A. Matrawy. A classification of web browser fingerprinting techniques. In *7th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2015. pages 48

N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, and V. Paxson. Header enrichment or ISP enrichment?: Emerging privacy threats in mobile networks. In *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, pages 25–30. ACM, 2015. pages 11

K. Waddell. Your Phone Is Listening—Literally Listening—to Your TV, 2015. URL https://www.theatlantic.com/technology/archive/2015/11/your-phone-is-literally-listening-to-your-tv/416712/. pages 17

J. Walthers. Learning to rank for cross-device identification. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference*, pages 1710–1712. IEEE, 2015. pages 9

Wikipedia. Wikipedia: Clustering coefficient, 2017a. URL https://en.wikipedia.org/wiki/Clustering_coefficient. pages 49

Wikipedia. Wikipedia: Likelihood-Ratio Test, 2017b. URL `https://en.wikipedia.org/wiki/Likelihood-ratio_test`. pages 46

Wikipedia. Wikipedia: Maximum Likelihood Estimation, 2017c. URL `https://en.wikipedia.org/wiki/Maximum_likelihood_estimation`. pages 26

Wikipedia. Wikipedia: Network Address Translation, 2017d. URL `https://en.wikipedia.org/wiki/Network_address_translation`. pages 3

G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *Security and Privacy (SP), 2010 IEEE Symposium*, pages 223–238. IEEE, 2010. pages 13, 14

W. Xie, C. Li, F. Zhu, E. Lim, and X. Gong. When a friend in twitter is a friend in life. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 344–347. ACM, 2012. pages 20

S. Zimmeck. *Using Machine Learning to Improve Internet Privacy*. PhD thesis, COLUMBIA UNIVERSITY, 2017. pages 9, 10

Appendices

Appendix A

Experiments Results

No.	Size	Clustering Coefficients	Total Num. URLs	Ave. Num. Deduced Employees	Av. TPR
1	15	0.649	12	1.0	1.0
2	20	0.444	12	1.0	0.97
3	40	0.249	12	1.0	0.83
			24	1.0	0.85
4	80	0.459	12	1.0	1.0
			24	1.0	1.0
5	94	0.546	12	1.0	1.0
			24	1.0	1.0

Table A.1: Performance of the *Top Likelihood Deduction* algorithm with different numbers values of *N*, for each social circle.

No.	Size	Clustering Coefficients	Total Num. URLs	Ave. Num. Deduced Employees	Av. TPR
1	15	0.649	12	2.356	1.0
2	20	0.444	12	5.87	0.69
3	40	0.249	12	4.54	0.606
			24	8.98	0.639
4	80	0.459	12	12.87	0.779
			24	14.76	0.776
5	94	0.546	12	14.01	0.939
			24	20.31	0.964

Table A.2: Performance of the *Eliminate & Iterate Deduction* algorithm with different numbers values of *N*, for each social circle.

No.	Total Num. URLs	Av. TPR		
Top % of Ranking		100%	50%	10%
1	12	1.0	1.0	1.0
2	12	0.69	0.846	0.95
3	12	0.606	0.734	0.79
	24	0.639	0.755	0.97
4	12	0.779	0.986	1.0
	24	0.776	0.925	0.994
5	12	0.939	0.981	0.944
	24	0.964	0.992	0.995

Table A.3: Performance of ranking the deduced Twitter accounts with different numbers values of *N*, for each social circle.

No.	Num. Anonymous URLs	*M*	Individual Av. TPR (Employees)	Group Av. TPR (Employees)	*M*	Av. TNR (Followees)	Av. TNR (Followers)	Av. TNR (Strangers)
1	1	5	0.8	1.0	50	0.0	0.02	0.12
	6		1.0	1.0		0.6	0.72	0.78
	12		1.0	1.0		1.0	0.96	0.98
	24		1.0	1.0		1.0	0.96	0.88
	36		1.0	1.0		1.0	0.96	0.96
	60		1.0	1.0		1.0	1.0	0.96
2	1	10	0.8	1.0	50	0.08	0.04	0.08
	6		1.0	1.0		0.2	0.22	0.76
	12		1.0	1.0		0.4	0.5	0.98
	24		1.0	1.0		0.92	0.96	0.98
	36		1.0	1.0		1.0	1.0	1.0
	60		1.0	1.0		1.0	1.0	0.98
3	1	17	0.765	1.0	50	0.06	0.08	0.22
	6		0.882	1.0		0.2	0.56	0.8
	12		1.0	1.0		0.28	0.62	0.96
	24		1.0	1.0		0.64	0.84	1.0
	36		1.0	1.0		0.88	1.0	1.0
	60		1.0	1.0		1.0	1.0	0.98
4	1	33	0.242	1.0	50	0.0	0.02	0.12
	6		0.788	1.0		0.24	0.36	0.72
	12		0.939	1.0		0.52	0.62	0.84
	24		0.939	1.0		0.8	1.0	0.86
	36		0.939	1.0		0.92	0.96	0.94
	60		0.939	1.0		1.0	0.98	0.94
5	1	33	0.333	1.0	50	0.1	0.02	0.22
	6		0.929	1.0		0.4	0.26	0.72
	12		0.976	1.0		0.76	0.76	0.76
	24		1.0	1.0		0.9	0.88	0.78
	36		1.0	1.0		0.9	0.88	0.86
	60		1.0	1.0		0.9	0.88	0.84

Table A.4: Performance of the algorithm to verify anonymous PMDs for each social circle. Tests are conducted with $M = 50$ maximum randomly selected users from each set of anonymous user type.

Adversary	Num. Anonymous URLs		Individual Av. TPR (Members)	Group Av. TPR (Members)	Av. TNR (Followees)	Av. TNR (Followers)	Av. TNR (Strangers)
None	1	score	0.357	1.0	0.04	0.06	0.26
	6	score	0.857	1.0	0.28	0.36	0.8
	12	score	0.905	1.0	0.46	0.6	0.82
	24	score	0.929	1.0	0.86	0.96	0.88
	36	score	0.929	1.0	0.92	0.96	0.8
	60	score	0.929	1.0	0.92	0.96	0.82
Network Eavesdropper	1	%	0.381	0.381	0.229	0.122	0.31
		score	0.119	0.381	0.02	0.02	0.08
	6	%	0.29	0.29	0.347	0.371	0.233
		score	0.357	0.714	0.0	0.06	0.02
	12	%	0.305	0.305	0.342	0.321	0.228
		score	0.476	0.857	0.22	0.3	0.26
	24	%	0.275	0.275	0.204	0.187	0.114
		score	0.595	0.833	0.2	0.36	0.3
	36	%	0.271	0.271	0.301	0.272	0.086
		score	0.667	0.833	0.8	0.84	0.36
	60	%	0.26	0.26	0.325	0.283	0.302
		score	0.69	0.857	0.9	0.94	0.82

Table A.5: Performance of the algorithm to verify anonymous PMDs when faced with the limitations of network eavesdroppers, for social circle no. 5. Tests are conducted with $M = 50$ randomly selected users from each set of user type.

Adversary	Num. Anonymous URLs		Individual Av. TPR (Members)	Group Av. TPR (Members)	Av. TNR (Followees)	Av. TNR (Followers)	Av. TNR (Strangers)
Google	1	%	0.357	0.357	0.438	0.612	0.262
		score	0.143	0.357	0.0	0.0	0.0
	6	%	0.343	0.343	0.448	0.418	0.23
		score	0.452	0.881	0.0	0.16	0.0
	12	%	0.316	0.316	0.403	0.352	0.226
		score	0.595	0.881	0.24	0.36	0.24
	24	%	0.323	0.323	0.233	0.207	0.118
		score	0.857	0.952	0.26	0.42	0.32
	36	%	0.314	0.314	0.359	0.317	0.09
		score	0.833	0.929	0.82	0.86	0.32
	60	%	0.319	0.319	0.384	0.332	0.347
		score	0.833	0.929	0.9	0.96	0.82
Facebook	1	%	0.19	0.19	0.125	0.204	0.143
		score	0.143	0.19	0.0	0.02	0.04
	6	%	0.167	0.167	0.25	0.252	0.215
		score	0.238	0.476	0.0	0.04	0.0
	12	%	0.143	0.143	0.165	0.151	0.121
		score	0.333	0.524	0.06	0.12	0.02
	24	%	0.136	0.136	0.097	0.095	0.064
		score	0.429	0.595	0.08	0.14	0.06
	36	%	0.136	0.136	0.235	0.219	0.078
		score	0.452	0.571	0.82	0.78	0.22
	60	%	0.14	0.14	0.243	0.227	0.269
		score	0.595	0.667	0.92	0.96	0.82
AppNexus	1	%	0.095	0.095	0.104	0.327	0.119
		score	0.071	0.095	0.0	0.02	0.04
	6	%	0.127	0.127	0.08	0.102	0.019
		score	0.095	0.476	0.02	0.1	0.08
	12	%	0.115	0.115	0.076	0.071	0.014
		score	0.262	0.667	0.1	0.14	0.08
	24	%	0.106	0.106	0.051	0.052	0.008
		score	0.429	0.81	0.14	0.24	0.12
	36	%	0.099	0.099	0.12	0.111	0.009
		score	0.476	0.762	0.66	0.64	0.16
	60	%	0.101	0.101	0.139	0.126	0.138
		score	0.595	0.833	0.82	0.9	0.8
ComScore	1	%	0.357	0.357	0.313	0.388	0.238
		score	0.119	0.357	0.02	0.02	0.04
	6	%	0.278	0.278	0.396	0.384	0.204
		score	0.262	0.786	0.0	0.08	0.0
	12	%	0.275	0.275	0.358	0.327	0.194
		score	0.476	0.881	0.14	0.34	0.18
	24	%	0.254	0.254	0.202	0.183	0.098
		score	0.643	0.952	0.18	0.36	0.22
	36	%	0.236	0.236	0.259	0.217	0.076
		score	0.667	0.929	0.66	0.7	0.3
	60	%	0.226	0.226	0.26	0.219	0.224
		score	0.69	0.929	0.82	0.86	0.76

Table A.6: Performance of the algorithm to verify anonymous PMDs when faced with the limitations of malicious third-party sites, for social circle no. 5. Tests are conducted with $M = 50$ randomly selected users from each set of user type.

No.	Ave. Num. Employees						Ave. TPR					
% of Private Accounts	0	20	40	60	80	100	0	20	40	60	80	100
1	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0
2	1.0	1.0	1.0	1.0	1.0	1.0	0.97	0.97	0.97	0.97	0.97	0.0
3	1.0	1.0	1.0	1.0	1.0	1.0	0.83	0.713	0.75	0.71	0.67	0.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99	0.04	0.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99	0.55	0.0

Table A.7: Performance of the *Top Likelihood Deduction* algorithm with different percentage of private Twitter accounts, for each social circle.

No.	Ave. Num. Employees						Ave. TPR					
% of Private Accounts	0	20	40	60	80	100	0	20	40	60	80	100
1	2.356	1.933	1.456	1.227	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0
2	5.87	5.86	5.69	6.27	5.83	6.33	0.69	0.684	0.643	0.466	0.252	0.0
3	4.54	4.52	4.52	4.14	4.12	3.44	0.606	0.593	0.588	0.411	0.357	0.0
4	12.87	12.34	12.1	10.23	10.16	9.9	0.779	0.745	0.736	0.607	0.156	0.0
5	14.01	11.72	10.98	10.08	8.19	6.43	0.939	0.887	0.737	0.645	0.455	0.0

Table A.8: Performance of the *Eliminate & Iterate Deduction* algorithm with different percentage of private Twitter accounts, for each social circle.

No.	Ave. Num. Employees					Ave. TPR				
% of Noise URLs	20	40	60	80	100	20	40	60	80	100
1	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
2	1.0	1.0	1.0	1.0	1.0	0.97	0.58	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0	1.0	0.83	0.0	0.0	0.0	0.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.78	0.0	0.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0

Table A.9: Performance of the *Top Likelihood Deduction* algorithm with different percentage of noise URLs, for each social circle.

No.	Ave. Num. Employees					Ave. TPR				
% of Noise URLs	20	40	60	80	100	20	40	60	80	100
1	2.356	13.12	17.12	24.12	25.12	1.0	0.162	0.124	0.088	0.084
2	5.87	16.87	20.87	27.87	28.87	0.69	0.24	0.194	0.145	0.14
3	4.54	15.54	19.54	26.54	27.54	0.606	0.177	0.141	0.104	0.1
4	12.87	23.87	27.87	34.87	35.87	0.779	0.42	0.36	0.288	0.28
5	14.01	25.01	29.01	36.01	37.01	0.939	0.526	0.453	0.365	0.355

Table A.10: Performance of the *Eliminate & Iterate Deduction* algorithm with different percentage of noise URLs, for each social circle.

No.	Ave. Num. Employees							Ave. TPR						
Num. Decoys	0	1	2	5	10	15	l	0	1	2	5	10	15	l
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.97	0.0	0.0	0.0	0.04	0.08	0.11
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.83	0.0	0.0	0.0	0.0	0.0	0.14
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.01

Table A.11: Performance of the *Top Likelihood Deduction* algorithm with different numbers of decoy Twitter accounts for each social circle, where $l \equiv |\text{social circle}|$.

No.	Ave. Num. Employees							Ave. TPR						
Num. Decoys	0	1	2	5	10	15	l	0	1	2	5	10	15	l
1	2.356	1.0	1.96	2.758	2.407	2.356	2.356	1.0	0.0	0.0	0.198	0.142	0.0	0.0
2	5.87	1.0	2.0	3.48	4.72	4.63	5.22	0.69	0.0	0.005	0.187	0.324	0.229	0.383
3	4.54	1.0	2.0	4.5	6.41	5.85	4.55	0.606	0.0	0.0	0.02	0.023	0.019	0.16
4	12.87	1.15	2.15	4.65	6.45	7.8	12.39	0.779	0.0	0.0	0.028	0.271	0.223	0.325
5	14.01	1.0	2.0	5.0	9.09	11.35	13.31	0.939	0.0	0.0	0.0	0.051	0.13	0.499

Table A.12: Performance of the *Eliminate & Iterate Deduction* algorithm with different numbers of decoy Twitter accounts for each social circle, where $l \equiv |\text{social circle}|$.