Revealing Privacy in Online Social Networks An exercise in revealing private friends

Benjamin Holland and Yong Guan
College of Electrical and Computer Engineering
Iowa State University, Ames, IA, 50011
Email:{bholland,guan}@iastate.edu

***Abstract*—With rising privacy concerns and the widespread adoption of social media platforms, it becomes necessary to examine how much information is still exposed after privacy protection mechanisms are enabled. Malicious attackers and investigators looking to utilize social media often face privacy and platform restrictions that serve both to protect the privacy of individuals and to protect the economic livelihood of the social media platform. This work focuses on revealing private social contacts as a first step to a larger reconnaissance of a target Online Social Network (OSN) profile. In many OSNs the friends that are hidden when a user enables privacy protection mechanisms may still be discovered by visiting the profiles of the target's friends and searching for public references to the target account. For large OSNs such as Facebook it would be unfeasible to search large portions of the network to discover friends of the target. In this work, we review existing Social Network Analysis (SNA) research to examine how it can be exploited to efficiently discover a majority of private friends. As our contribution, we propose a greedy search algorithm for enabling efficient discovery of private friends on social networking sites, which has shown a 33.6% to 41.6% improvement over a standard breadth-first search on synthetically generated social networks as real-world social networks.**

I. Introduction

From the perspective of an investigator or cyberstalker an Online Social Network (OSN) can potentially offer valuable information concerning a target. OSNs contain status updates, personal pictures, hometown information, religious and political views, relationship information, and most importantly social connections. When investigating a target one of the first steps is to discover the target's friends and social contacts, but with increasing concerns over privacy many users have begun to enable privacy protections that obscure social contacts and mask personal information. What many users don't realize however is that their online digital footprint is often larger than they expect. By searching the neighboring nodes of the target that do not enable privacy protection mechanisms, the attacker can still discover the target's social connections by searching for friends that publicly declare a connection to the target. Once the attacker has collected a list of previously obscured friends the second step is to make a closer inspection of each friend's profile, photos, and online interactions that may reveal personal details of the target. In this paper we discuss the first step of this attack and focus on an algorithm for efficiently uncovering private friends in a social graph with privacy protections.

This attack has practical purposes in the Open Source Intelligence (OSINT) community by providing investigators with a tool to aid in background investigations, validating alibis, or creating pattern-of-life matrices. In this study we found that we could discover private friends on social networking sites with a 33.6% to 41.6% improvement over a standard breadth-first search on synthetically generated social networks as well as major improvements on real-world social networks.

A. Privacy Restrictions

With growing privacy concerns, many social networking platforms have continued to add privacy control mechanisms to restrict access to private information. As of May 2010, Facebook offered an excess of 170 privacy settings and maintained a privacy policy longer than the U.S. Constitution [14]. Despite growing privacy concerns and public calls for increased legislation to enforce the protection of individual privacy, a late 2010 crawling of Facebook revealed that only about a quarter of Facebook users enable privacy protections that restrict access to their friend lists [8]. In a 2008 telephone based survey, it was found that approximately 50% of adults and more than 75% of teens thought it would be difficult or impossible to find out who they were based on the information publicly available in restricted social networking profiles [12], which may indicate a false sense of privacy when dealing with OSNs. A more recent survey conducted in early 2012, shows that profile pruning (removing personal information) and unfriending contacts in the interest of privacy is on the rise [13].

B. Platform Restrictions

Information flowing into online social networks is collected on a massive scale, but is tightly controlled by the social media platform regarding how it flows out. Social networking platforms generally control information flowing out based on social relationships, user-based privacy settings, as well as rate limiting, activity monitoring, and IP address based restrictions. While many access control mechanisms are often put in place to protect the privacy of the user, other mechanisms are often added or intentionally handicapped to protect the economic livelihood of the service platform.

C. Data Availability

In the worst-case scenario for the attacker, the desired information may have never been gathered by platform, such

as when a user chooses not to provide information on a social networking profile. If the desired information was never recorded, an attacker has no hope of discovering the information. In a more typical case, the target information exists, but privacy and platform restrictions introduce a fog that masks large portions of the desired information. As a result, the attacker is forced to develop techniques of exploring the fog and examining the digital footprints left behind by user interactions.

D. Data Longevity

The social graph is far from static, relationship dynamics change frequently and profiles are updated constantly. Facebook claims that of its approximately 850 million users, over half its users log in at least once a day [16]. Previous studies of the Facebook social graph have limited collection periods to a maximum of one to two weeks (depending on the type of data) [20] [7] to limit the corruption of data due to changes in the social graph during collection periods. Monitoring the changes of social content is also an important source of information for understanding the dynamics of the social graph. We can think of each data access as a snapshot in time that is capturing the state of the social graph at collection time.

In Section 2 we discuss some preliminary works that review common properties of social networks that are useful to understanding the algorithm we later propose for finding private friends. In Section 3 we formally state the problem definition and outline some intuitions. Section 4 describes our proposed algorithm for discovering private friends. In Section 5 we discuss our results and then we provide some concluding remarks in Section 6.

II. Preliminaries

There are many metrics used to describe graph structures, but when dealing with social networks researchers tend to describe social graphs in terms of size, degree distribution, average shortest path, and clustering. This section briefly reviews each metric in order to build a foundation for the rest of this work.

First, we define an undirected graph $G = (V, E)$ to be a set of vertices (nodes) V connected by the set of edges (relationships) E. For a given pair of vertices A and B we denote the existence of an undirected edge between A and B with the notation $< A, B >$, where $< A, B >=< B, A >$.

A. Size

The number of nodes $n = |V|$ in a graph usually defines the size of a social network. Consequently, size is an exemplary metric for expressing the scope of information contained in a graph. Managing large social graphs with many relationships can be very resource intensive. A 2010 estimate of the overhead required to crawl the Facebook social graph was roughly 44 Terabytes of data [7]. Another useful metric is the number of edges in the graph. An undirected graph can have at most $\frac{n(n-1)}{2}$ edges. With the set of edges E and the set of vertices V we can calculate graph density (a measure that represents the ratio of the number edges to the maximum number of edges in an undirected graph) for an undirected graph as:

$$density = \frac{|E|}{\frac{|V|(|V|-1)}{2}} = \frac{2 * |E|}{|V|(|V|-1)} \tag{1}$$

B. Degree Distribution

Aside from knowing the number of nodes in a network, we often want to describe the distribution of friends (node degrees) among all nodes in the network. A degree is defined as the number of adjacent neighbors connected to a given node. Many social networks have a degree distribution that asymptotically follows a power-law distribution (also known as a Pareto distribution) as in the following relation (where k is the degree and λ is a variable parameter usually $2 < \lambda < 3$) [9]:

$$P(k) \sim k^{-\lambda} \tag{2}$$

Graphs with degree distributions that follow a power-law are called scale-free networks and are commonly explained using a preferential attachment model. In a preferential attachment model the rich (nodes with high degrees) get richer (gain additional degrees faster) than the poor (lesser degree) nodes [5] [4]. This property leaves a signature linear plot on a log-log scale.

C. Average Shortest Path Length

The average shortest path length is the average of the minimum length path (also know as a geodesic) required to connect a given pair of nodes for each node pair in a connected graph. When $n = |V|$ is the number of nodes in the graph and $d(v_s, v_t)$ is defined as the shortest distance from a given pair of vertices v_s to v_t (distance is 0 if $v_s = v_t$) the average shortest path is defined as:

$$L = \frac{1}{n(n-1)} \sum_{\forall v_s, v_t \in V} d(v_s, v_t) \tag{3}$$

In social networks, a small-world graph is a graph that has an average shortest path length that scales proportionally with the logarithm of the number of nodes in the graph [19].

$$L \propto \log n \tag{4}$$

This small-world phenomenon is well observed in social networks and is sometimes referred to by the six degrees of separation concept that is associated with the work done in the Stanley Milgram small-world experiment [18].

D. Clustering

Social networks tend to form definitive clusters of nodes that are made up of tight knit groups of highly connected nodes [19]. Supposing that a vertex V_i with edges E_i has f friends (neighbors), it follows that a node can have at most $\frac{f(f-1)}{2}$ edges. The local clustering coefficient C_i for a vertex i is similar to density in that it is defined as the ratio of edges that actually exist out of the set of possible edges a vertex can have [6].

$$C_i = \frac{2 * E_i}{f(f-1)} \tag{5}$$

The average clustering coefficient for the entire graph is defined as:

$$C = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{6}$$

III. Problem Definition

The goal of this study is to find the most (if not all) private friends of a target of interest by using a limited number of queries to search online social media websites. Most online social networks provide an Application Program Interface (API) to interact with the social graph and query information about a given node. A private node is a node that returns little to no information due to access restrictions imposed by a user. As a result a list of the target's friends may be obscured from the public view when accessing a node directly, but target friendships can still be verified by visiting the public friends individually.

Social media websites impose global access restrictions through rate limiting, access request quotas, and account activity based banning making API calls a limited resource. Due to the sheer size of most social networks it is unfeasible for attackers to brute force guess the friends of a target. Based on these metrics, we propose an algorithm for the efficient discovery of private friends (private nodes) in Section 4.2.

Formally, we define an evaluation metric of one API call to be the cost of expanding a single node to return the node's neighbors. Furthermore, a private node is defined as a node that returns an empty neighborhood set $\emptyset$, whereas a public node returns a non-empty neighborhood set. We define a target v_t to be a private node such that $v_t \in V$ in the undirected graph $G = (V, E)$. Vertex v_t contains f hidden undirected edges E_t corresponding to each friend (neighbor of v_t) represented by the set F. A subset $F_{pub} \subset F$ is the set of target friends that publicly disclose an edge linking from v_i to v_t. That is, the intersection of $E_i \cap E_t \neq \emptyset$ is not empty, which means it contains one shared edge with the target node v_t. Finally, we define a starting vertex $v_s \in V \neq v_t$ from which to begin searching the graph. An efficient search is a search that discovers most (if not all) nodes in F_{pub} that link to the target node v_t using the fewest API calls.

A. Intuitions

Consider this scenario. Bob, a high school student, has ten friends at school but won't tell us who his friends are. Even though Bob won't say who his friends are, we can still find the same information by walking around the school and asking students if they are friends of Bob. Assuming every student at the school either tells the truth or refuses to answer the question, we can exploit known social structures of the school to create an efficient search strategy to reveal all of Bob's friends.

Social graphs tend form large numbers of triadic closures of mutual friends [15], which means that if we ask Bob's best friend Jim (or any of Bob's friends for that matter) who their friends are, we are likely to find a set of friends that are mutually shared between both Bob and Jim. Likewise, it is more likely that a friend of Jim is also a friend of Bob, as opposed to another randomly selected individual. We can exploit the mutual friendship property of social graphs by searching for mutual friends of a newly discovered friend and the target. We use the number of times an individual has been observed as a friend-of-friend of the target as a metric to describe the priority in which we should interrogate friends-of-friends.

Social graphs also have strong clustering properties [19], so let's now consider that the school is made up of various clusters of students (e.g. the math team, the chess club, the football team, the band, etc.) and that we know Bob is the quarterback of the football team. It makes sense that we would start by asking members of the football team if they are friends of Bob because the members of the football team are more likely to be friends with Bob. The members of the football team are more likely to be friends with Bob because of the tight clustering of mutual friends between the football team clique. For the members of the math team, which share most of their mutual friend connections with other members of math team, a member is less likely than a member of the football team to make connections to members in the football team cluster. That's not to say that Bob and Mike (a member of the math team) can't be friends, it just that Bob is more likely to have more friends on the football team than the math team.

Let's pretend that we don't know that Bob is the on the football team. It makes less sense to interrogate the entire math team to see if they are friends with Bob than if we picked a few representative students from each clique to ask if they are friend's of Bob. The goal is to spend less time searching clusters that Bob is excluded from and more time searching clusters in which Bob is included. When we don't know what cluster Bob belongs to, it's better to cast out a wide net and interrogate individuals from each cluster (the math team, the football team, the chess club, the band, etc.) until we find a cluster that includes Bob.

In order to maximize the number of new nodes discovered with each node expansion we should choose to expand nodes that we have seen the fewest times first. On average, unexpanded nodes that have been observed more reveal less new nodes because the more a node has been observed the more neighbors of a node have been previously discovered. In the context of mutual clusters of friends expanding lesser-observed nodes has an effect of crawling away from clusters towards other clusters. Reversing the priority biases the crawl order towards staying within a cluster until the entire cluster has been explored. In our proposed algorithm we use this as a secondary heuristic to maximize search potential by breaking out of clusters that don't contain friends of the target.

IV. Hunter-Seeker Algorithm for Private Friend Discovery

A. Data Collection

In order to search for private friends the social graph must be crawlable. On the professional networking site LinkedIn crawling is not possible because all contacts are considered

private. Other networks such as Twitter and Facebook provide API's to assist in querying the networks. In the case of Facebook, the API is limited to the scope of the authenticated account's social network, preventing full graph traversals in both the old Facebook REST API and the newer Graph API. As a proof of concept for our attack, we wrote our own Facebook crawler using the Apache Commons HTTPRequest package and the Jericho HTML Parser to authenticate and make raw HTTP requests to crawl the Facebook social graph by navigating the HTML web interface. To avoid obstacles that make it difficult to screen-scrape such as JavaScript and AJAX requests we navigate to the mobile version of the Facebook interface that returns the same content in simple HTML.

B. Algorithm Design

Listing 1. Hunter-Seeker Algorithm

```
friends = map(origin, target){
  if(origin == target || origin.is_private)
    return error // nothing to work with
  pq.enqueue(origin, PRIORITY_0)
  neighborhood, searched = {}
  while(pq.size > 0){
    node = pq.dequeue()
    if(searched.contains(node)){
      continue // already explored node
    } else {
      neighbors = node.expand()
      if(neighbors.contains(target)){
        neighborhood.add(node)
        pq.enqueue(neighbors, PRIORITY_1)
      } else {
        pq.enqueue(neighbors, PRIORITY_0)
      }
      searched.add(node)
    }
  }
  return neighborhood
}
```

Given the information available to the algorithm at runtime, the algorithm behaves in one of two modes: hunter or seeker. In hunter mode, the algorithm has expanded a friend of the target and searches the friend's friends discovered for mutual friends of the target (giving the most discovered nodes of a known target friend search priority). In seeker mode, the algorithm has no friends-of-friends left to explore and attempts to maximize its potential of finding a new friend of the target by prioritizing the exploration of nodes expected to reveal the highest number of new nodes.

The private friend discovery algorithm (nicknamed Hunter-Seeker) shown in Listing 1 uses a priority queue with the properties listed below to manage the order of node exploration.

- Items in the priority queue are sorted by two scores (a primary score and a secondary score)
- A primary score for an item increases each time the item is added to the queue by a value defined on each add
- A secondary score for an item increases by one each time the item is added to the queue
- A higher primary score takes precedence over a lesser primary score
- A non-zero primary score always takes precedence over a secondary score
- A lower secondary score is used to break ties between the primary scores
- A tie between secondary scores is broken by order of arrival in the queue

An example run of the algorithm is provided on the toy network shown in Figure 1. The corresponding search tree steps taken by the Hunter-Seeker algorithm are shown in Figure 2 and traced in Table I. As the search tree is expanded unsearched nodes that are discovered to be a friend-of-friend of the target will have their primary score values incremented by one. Every time a node is observed in the neighborhood of another node, the secondary priority is incremented by one. Scores are only tracked for nodes that have not been searched. The target node is never searched because it is assumed to be private and would not have any affect on the queue order.

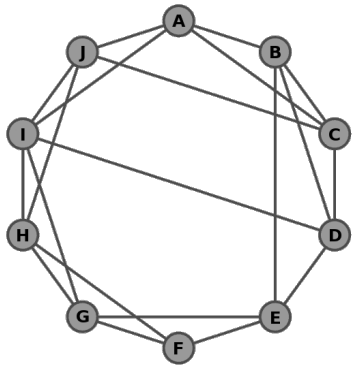

Fig. 1. Sample Toy Network

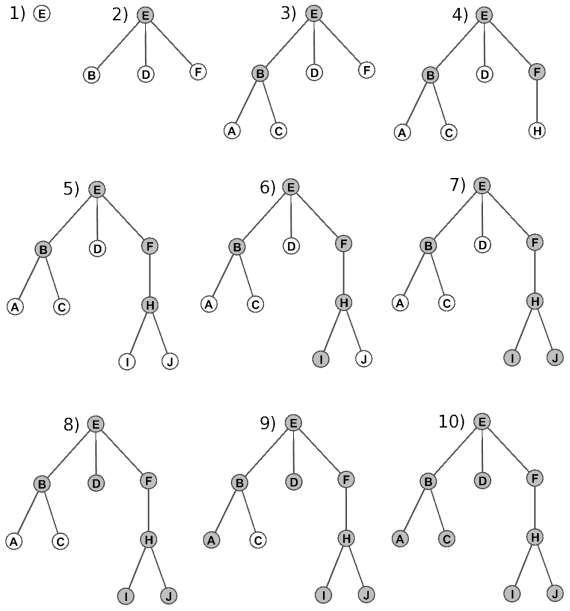

Fig. 2. Toy Network Search Tree for Hunter-Seeker Algorithm

TABLE I
HUNTER-SEEKER ALGORITHM ON TOY NETWORK (START NODE: E, TARGET NODE: G)

Step	Searching	Priority Queue {Node , Primary Score Value : Secondary Score Value}	Friends Discovered
1	E	[{E,0:1}]	[]
2	B	[{B,1:1}, {D,1:1}, {F,1:1}]	[E]
3	F	[{F,1:1}, {D,1:2}, {A,0:1}, {C,0:1}]	[E]
4	H	[{H,1:1}, {D,1:2}, {A,0:1}, {C,0:1}]	[EF]
5	I	[{I,1:1}, {J,1:1}, {D,1:2}, {A,0:1}, {C,0:1}]	[EFH]
6	J	[{J,2:2}, {D,2:3}, {A,1:2}, {C,0:1}]	[EFHI]
7	D	[{D,2:3}, {A,1:3}, {C,0:2}]	[EFHI]
8	A	[{A,1:3}, {C,0:3}]	[EFHI]
9	C	[{C,0:4}]	[EFHI]

C. Analysis

1) Optimal Friend Discovery: In the optimal case, a perfect algorithm would discover every private friend using the absolute minimum required API calls in the process. If we say that the target has f friends, then we find that for each node in the friends list we must make one API call to confirm the friend is indeed a friend. We must then add to f the minimum number of nodes required to reach each friend from a given start point to cover the optimal case. The minimum number of nodes is defined as the minimum spanning tree that includes the start node and each of the required friend nodes. The rest of the nodes may optionally be included in the minimum spanning tree if they are required to complete a path between required nodes in the tree. This problem is classified as the node-weighted Steiner tree problem [10] and is computationally expensive to compute.

For the purpose of simplicity we will assume the optimal case is defined by an algorithm that is either extremely lucky or has access to an all knowing oracle that reveals the friend list to the search agent. Upon learning the friends list, the search agent simply needs to confirm each friend at a rate of one friend per API call. On a scatter plot like the plots shown in Figure 6 the optimal case would be a nearly vertical sloped line with the x coordinate starting at $\frac{1}{n}$ API calls and moving to $\frac{f}{n}$ API calls and the y coordinate starting at $\frac{1}{f}$ friends ending at $\frac{f}{f} = 1$ friends. A successful algorithm implementation is that one that approaches the optimal case once a target friend is discovered and on average does better than BFS, which is the standard search method for crawling social networks at this time.

2) General Case: Given what we know of social networks, we assume that the probability that an edge exists between nodes A and C given that there exists a pair of edges between nodes A and B and nodes B and C is more likely than the probability that an edge exists between a pair of randomly selected nodes. This assumption is supported by observations of triadic closures and clustering in real-world social networks as discussed earlier.

$$P(< A, C > \mid < A, B > \ and \ < B, C >) > \frac{1}{2} \quad (7)$$

The base case structure that our algorithm exploits is shown in Figure 3. In the base case, node A is the target and the algorithm starts searching at node D. A BFS would discover

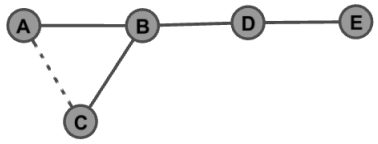

Fig. 3. Base Case Structure

nodes in the order of DBEC giving priority to node E over node C because it was discovered earlier. If we consider the distance of a node to be the shortest path that connects the target and a node, we find that all friends are located distance 1 from the target. When our algorithm searches node B it discovers that B is a friend of the target and that node C is at most distance 2 from the target. Before node C is searched, it is unknown if there is a path of distance 1 from the target A, but from the triadic closure properties of social networks we know that there is a better than random chance that a path of distance 1 does exist. Node E is located at distance 3 and statistically has less chance of being connected to node A than node C, so it is better to give priority to node C. Nodes of distance 3 or greater (at runtime) are never given search priority. Our algorithm searches the base case structure in the order of DBCE.

While a BFS is not random, we have found experimentally that the average probability that the next node in a BFS queue is a friend of the target is comparable to a randomly ordered search queue (shown in Figure 4). The Hunter-Seeker algorithm exploits non-random properties in social networks, which leads us to believe that our algorithm should outperform a BFS. To support this claim we offer the following logical steps.

Step 1) Assume the following graph property:

$$P(< A, C > \mid < A, B > \ and \ < B, C >) > \frac{1}{2}. \quad (8)$$

Step 2) Let,

(a) f be the number of friends of the target
(b) n be the number of nodes in the graph
(c) λ represent the probability that the next node in the search queue is a friend of the target for the i^{th} friend discovered.

Step 3) Regardless of the algorithm, the probability of a node being a friend of the target is $\frac{f}{n-1} \approx \frac{f}{n}$.

Step 4) For a randomly filled search queue, the probability of the next node in the queue being a friend is $(\frac{f}{n-1})(\frac{f-1}{n-2}) \approx \frac{f^2}{n^2}$. The probability of i next nodes in the queue being friends is $\prod_{0}^{i} \frac{f-i}{n-i-1} \approx (\frac{f}{n})^2$, where $i < f$. The probability of finding a friend after discovering all f friends is zero.

Step 5) The approximate number of nodes searched to find a friend is $n * \frac{1}{f-1} \approx \frac{n}{f}$. The approximate number of nodes searched to find the i^{th} friend is $\frac{n(i)}{f}$. Therefore the average case of λ for a random search of the graph is defined by $\frac{f-i}{n-\frac{n(i)}{f}-1}$, where $1 \leq i \leq f$.

Step 6) Experimentally, we determine that the average case for a BFS on a graph with the assumed properties approximates a random search of the graph in terms of λ. Figure 4 shows the calculated values for a random search (which we have verified experimentally) and an average of 500 runs of a BFS on randomly selected start and target nodes from a 1000 node Watts-Strogatz graph with an average 130 friends. At runtime, after each new friend is found, λ is computed by calculating $\frac{f-f_f}{n-s-1}$, where f_f is the number of friends found and s is the number of nodes searched. The results show that, on average, a BFS is comparable to a random search.

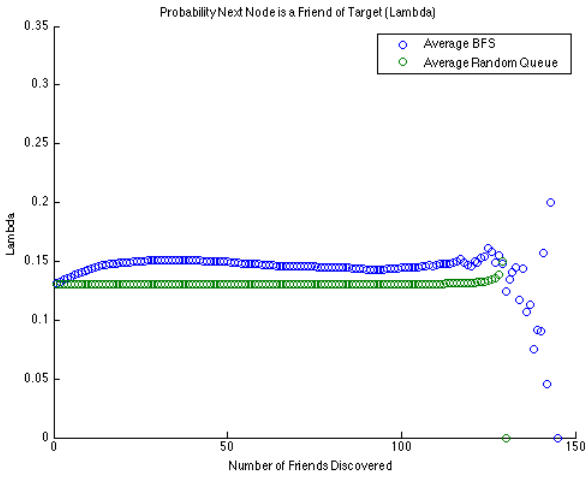

Fig. 4. Comparison of BFS to Random Search

Step 7) The Hunter-Seeker algorithm exploits the probability that a friend-of-friend relationship has a λ value greater than $\frac{1}{2}$ for nodes neighboring a friend by frontloading friend-of-friend nodes in the queue. We can therefore assume that while the majority of friend nodes exist the Hunter-Seeker algorithm will maintain a i^{th}-λ probability of $(\frac{f}{n-1})(\lambda)^i \approx (\frac{f}{n})(\lambda)$, where $\lambda > \frac{1}{2}$.

Finally, we can show that the Hunter-Seeker algorithm is better than a random search, which is comparable to a BFS, by showing that $(\frac{f}{n})^2 < (\frac{f}{n})(\lambda)$ because $\frac{f}{n} << \lambda$. Note that because the Hunter-Seeker algorithm places priority on searching nodes more likely to be friends earlier in the search, the algorithm is left with fewer friends to find later in the algorithm execution. The result is that after about 80% of the friends are discovered the performance of the algorithm begins to decline (shown later in our Results Section).

3) Worst Case:

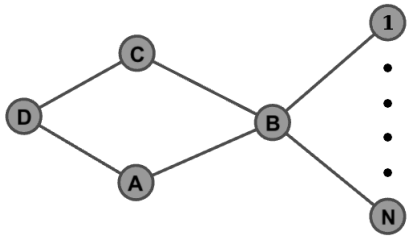

Fig. 5. Worst Case Structure

In the worst case, our algorithm searches a structure like the one shown in Figure 5. If the target node is A and the starting node is node C a BFS could add nodes B and D to the search queue (order is completely arbitrary for BFS). Once node B is searched, a BFS and our algorithm diverge as the Hunter-Seeker algorithm gives priority to nodes 1 through N, which are all bad guesses. In this case, the algorithm is led astray by the promise of mutual friends of the target and node B while a BFS outperforms by giving priority to node D over nodes 1 through N. This case contradicts our basic assumption of triadic closures because in this graph $P(< A,C > \mid < A,B > \ and \ < B,C >) = 0$, but we know that triadic properties do exist in online social networks from empirical observations [15]. Through the law of averages we know that even though the worst case structure shown in Figure 5 may exist as a subgraph, there must be more cases like the structure shown in Figure 3 that our algorithm successfully exploits for the triadic closure property to be observed. The result is that our algorithm performs better than a BFS in an average case for graphs with social network properties.

V. RESULTS

In this section, we show the evaluation results of both synthetic data and real-world data.

A. Test Framework

To generate synthetic data, we assume that online social networks such as Facebook have properties similar to the Watts-Strogatz small-world random graph model. Previous works [19] [7] have show that the average shortest path length and clustering coefficients of real-world social graphs can be reasonably well modeled by the Watts-Strogatz small-world graph. It is known that the Watts-Strogatz model does not reproduce the power-law degree distributions found in most observations [9] of social networks, but at the time of this writing we could not find a random graph model that could accurately represent the set of all properties observed in real world social networks. Our algorithm leverages mutual friend relationships and group clique formation that we find present in online social networks such as Facebook [11], which of the models we evaluated (Erdős-Rényi Model, Watts-Strogatz, and

Barabási-Albert), the Watts-Strogatz model demonstrated best. To evaluate our algorithm's performance we run it on datasets generated using the Watts-Strogatz random graph model and later verify our model results by comparing the results of algorithm on real-world social networks.

We generate Watts-Strogatz random graphs using the advanced graph model generator plugins available for the Gephi graph visualization utility (*http://www.gephi.org*). We then import the models into our test platform, which simulates the role of an online social network by restricting the set of node neighbors visible for a given node based on a single privacy restriction that can be enabled or disabled for each node. At run time we import a new random graph, enable privacy for the target node and a variable percentage of the remaining nodes, and run a test harness. The test harness chooses two random start locations (one in the set of the target's friends, and one outside the set of the target's friends) and records the results of a BFS, DFS, and our algorithm for both types of starting locations. The test harness is run over twenty or more iterations where each iteration records a pair of data points for each new friend discovered. The first data point corresponds to the ratio of the number of API calls used to the total number of nodes in the graph (the number of API calls required to search every node in the graph). The second data point is the ratio of the number of friends discovered to the actual number of friends connected to the target node. After each iteration, privacy settings are reset to prevent skewing the results of later runs of the algorithm on the same dataset. The results are then serialized to a file and exported to Matlab to be analyzed as histograms and scatter plots.

B. Comparison to Existing Algorithms

To verify that our algorithm indeed works in a base case scenario (i.e. when all nodes are public except for the target), we tested our algorithm on a small test network of 1000 nodes generated using Watts-Strogatz model parameters of $G(n, k, p) = G(1000, 130, .2)$. We justify our average degree $k = 130$ by equating it to the average number of friends a user has on Facebook (a statistic shared by Facebook during their F8 Developer Conference). We justify a random rewiring probability value of $p = .2$ by intuitively reasoning that less than half of friendship connections on Facebook are comprised of random connections and because $p = .2$ generates a model with an expected average clustering coefficient of $C = .384$, which is within the range observed by previous work [11].

Examining the scatter plot results (shown top-left and middle-left in Figure 6) shows that our algorithm using the Hunter-Seeker strategy consistently performs better than both BFS and DFS regardless of the starting location. When the starting location is moved outside of the target neighborhood both BFS and DFS are negatively impacted in terms of API calls per friends discovered, but the Hunter-Seeker algorithm remains relatively unaffected.

Histogram plots of the same graphs for the Hunter-Seeker algorithm show the number of friends revealed in the first percentages of the number of API calls to the total number

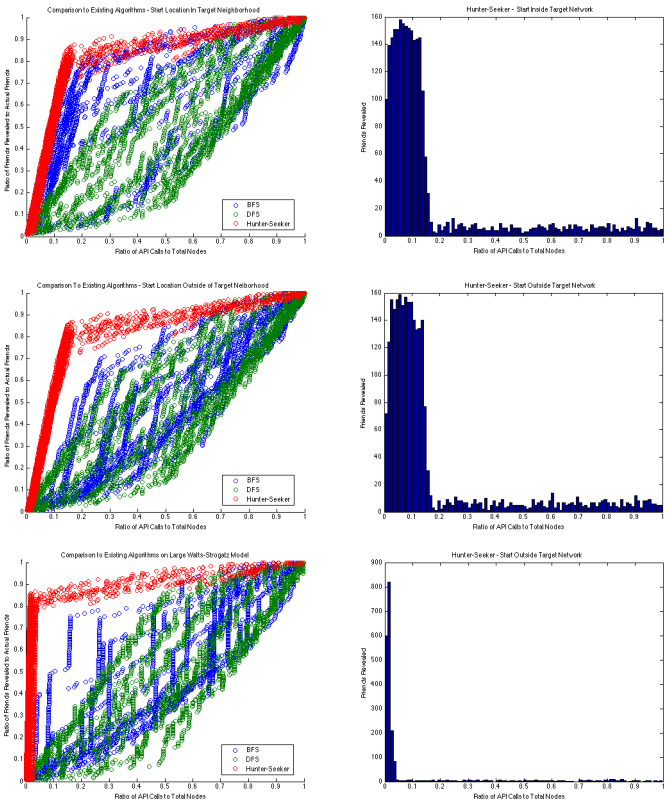

Fig. 6. Performance of Hunter-Seeker algorithm on Watts-Strogatz random graph models. Note: The scatter plot graphs should be viewed in color.

of nodes includes the majority of all friends. We find that the algorithm finds most of its target's friends in the first 10-20% of the total nodes and then spends the rest of the algorithm searching for the edge nodes that were not near the target and most likely located in other clusters. At this point in the algorithm, there is a surplus of bad guesses the algorithm could make and very few right answers. A comparison of both histograms confirms that the algorithm will produce similar results for a random location in the graph versus a start location inside the target neighborhood.

To compare our results to a larger graph we scaled the graph by creating a Watts-Strogatz model with parameters $G(10000, 130, .2)$. The resulting graph took about 24 hours to complete the test harness iterations of each algorithm on a modern i7 laptop processor. The graph contained $\frac{nk}{2} = 650000$ edges. Figure 6 (bottom) shows that as n increases the number of nodes ignored by the Hunter-Seeker algorithms also increases, widening the performance difference between BFS and DFS and the Hunter-Seeker algorithm relative to n. By integrating the area under the trend line curves of each algorithm, we find that the Hunter-Seeker algorithm makes a 33.6% to 41.6% improvement over a standard breadth-first search on the 1000 and 10,000 node networks respectively.

C. Real World Networks

Perhaps the largest criticism of our results is that our random graph models do not account for free-scale degree distributions

that are commonly found in online social networks. The lack of random models that demonstrate free-scale distributions and the small-world and clustering coefficient properties as well as our inability to gather real-world results of our own (for legal reasons) has left us ill-equipped to fully address this concern. However, the results of our algorithm's performance on similar real-world networks discussed in this section leads us to believe that our strategy is viable in online social networks.

After speaking with our University Institutional Review Board (IRB) advising committee we were advised not to proceed with a test that would break the terms of service agreements defined by Facebook (and most other online social networks). Instead we looked for existing social network research that made public similar social network datasets. We found datasets collected as part of a study on face-to-face interactions in primary schools [17], trust networks [2], and paper citation [3] and collaboration networks [1], which are shown in Figure 7. These dataset contained mutual friend relationships and clustering that we intuitively designed our algorithm around, but we found that in some cases the context of network was fundamentally different. One such example is the Epinions online directed trust network, which we treated as an undirected friendship network for the purposes of this study. The results of our algorithm shown bottom left in Figure 7 seem to suggest that directed trust relationships do not translate well to undirected friendship relationships, meaning the attacker should pay special attention to the context of the network during analysis.

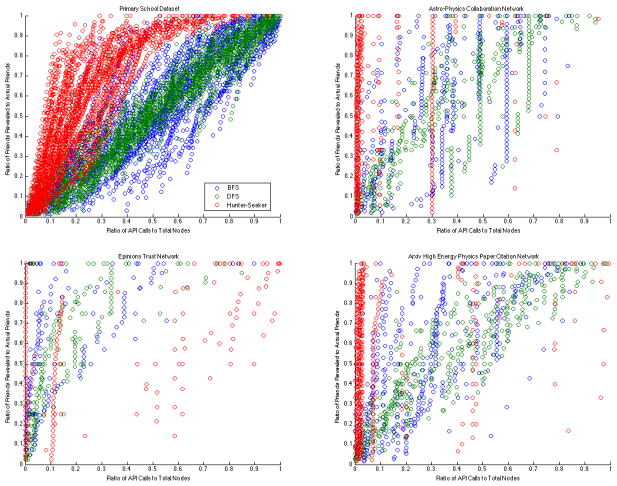

Fig. 7. Performance on Real-World Social Networks (Top Left: Primary School Face-To-Face Network [17], Top Right: Astro-Physics Collaboration Network [1], Bottom Left: Epinions Trust Network [2], Bottom Right: High Energy Physics Paper Citation Network [3]). Note that these graphs are best viewed in color.

D. Private Networks

To test our algorithm on private social networks we modified our test harness to randomly privatize a percentage of the total nodes (including the target) before selecting a start location. The test harness then picked two start locations, one that was inside the target network and one that was outside the target network to begin the test data collection process like before. We privatized nodes at 25%, 50%, and 75% on a $n = 1000$ node Watts-Strogatz random graph with parameters described previously and ran the test harness. In the average case our algorithm can discover n minus the number of privatized friends, which is on average $f*25\%$, $f*50\%$, and $f*75\%$ respectively, where f is the average node degree (average number of friends). The results are shown in Figure 8. The algorithm is capped at these bounds because there is no way to verify that a private friend and the target are friends since both nodes are private.

As future work we aim to incorporate a third metric, the distance from the target. Due to the fact that there is no way to verify if a private node is a friend of a target (that is also private) the upper bound is capped on the number of friends we can discover, but some metrics may be useful to denote nodes as likely but unverifiable friends of the target. In our current implementation, we tried returning the discovered friends along with a set of observed high-ranking friend-of-friend nodes that could not verified as friends of the target as likely friends, but there were far too many false positives to make the set useful. With the addition of a third metric (the distance from the target) it may be possible to reduce the number of false positives and break through the upper bound cap on private friend discovery.

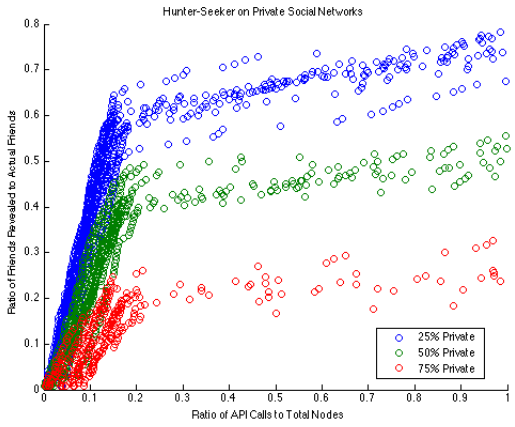

Fig. 8. Hunter-Seeker on Private Social Networks

VI. Conclusion

A. Summary

Investigators and cyberstalkers looking to discover private social contacts of a target in online social networks are met with privacy and platform restrictions as well as data availability and longevity issues. In this work, we reviewed the common properties of online social networks and devised an algorithm that exploits social network topologies to efficiently discover private friends of a target.

Our attack addresses privacy restrictions by discovering private friends of a target by exploiting mutual friend rela-

tionships and graph clustering properties. To address platform restrictions we defined a platform based metric of API calls to measure the algorithm's performance, which motivated us to optimize the algorithm to perform efficiently under practical constraints. While there is little we can do to change the availability of information, our algorithm does address data longevity issues by serving as a mechanism to enable efficient and automatic crawls of the social graph at times defined by the attacker. While previously it would have been found infeasible to search for a target's friends, we have proposed a practical, robust, and cost efficient algorithm to selectively search for private friends. By doing so we have shown that an attacker may be able to reproduce a target's friend list despite privacy protection mechanisms. Meanwhile, this attack may be a useful tool for Open Source Intelligence (OSINT) analysts in law enforcement fields.

We believe the implications of this work will primarily impact social network users. It is unclear as to what a reasonable expectation of privacy is and is not on online social networking sites, but we can safely assume that users that enable privacy protections are expecting at least some level of privacy protection from the social networking platform. To prevent this attack, users must enable privacy settings that remove their profile from the public completely (including a friend's friend list), but the result of doing so is likely to negatively impact the user's social interactions by limiting discovery by future contacts. We believe that most users are more likely to risk revealing private information than compromising social interactions, which is the primary motivator behind social networking sites, so the attack is difficult to prevent on a large scale.

B. Discussions

We would like to expand our framework to study cross-correlations between social networks. We believe that it may be possible to utilize a second reference network or set of reference networks to infer information that is not present in a single network through the use of graph de-anonymization techniques. A reference network may be gathered by examining an auxiliary social network or by saving portions of the primary social network from previous algorithm runs to enhance the performance of subsequent runs.

Finally, now that the groundwork has been laid to discover private friends it would be advantageous to examine the next stage of the attack and look at what can be learned about a target through the information gleaned from neighboring profiles. We believe that a wealth of information about a target exists in messages, pictures, and interactions of friend profiles that the target may be unaware of or unable to control.

REFERENCES

[1] Astro physics collaboration network. http://snap.stanford.edu/data/ca-AstroPh.html, 2003.

[2] Epinions social network. http://snap.stanford.edu/data/soc-Epinions1.html, 2003.

[3] High-energy physics citation network. http://snap.stanford.edu/data/cit-HepPh.html, 2003.

[4] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, January 2002.

[5] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, pages 509–512, September 1999.

[6] B. Bollobás, R. Kozma, and D. Miklós. *Handbook of large-scale random networks*. Springer Verlag, 2009.

[7] S. Cantanese, P. D. Meo, E. Ferrara, G. Fiumara, and A. Provetti. Extraction and analysis of facebook friendship relations. Technical report, University of Messina, Italy and University of Oxford, UK, 2010.

[8] S. Catanese, P. D. Meo, E. Ferrara, G. Fiumara, and A. Provetti. Crawling facebook for social network analysis purposes. *CoRR*, abs/1105.6307, 2011.

[9] A. Clauset, C. Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *ArXiv e-prints*, June 2007.

[10] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, January 1986.

[11] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. A walk in facebook: Uniform sampling of users in online social networks. *CoRR*, abs/0906.0060, 2009.

[12] P. Internet and A. L. Project. Adults and social network websites. http://www.pewinternet.org/Reports/2009/Adults-and-Social-Network-Websites.aspx, January 2009.

[13] M. Madden. Privacy management on social media sites. http://pewinternet.org/Reports/2012/Privacy-management-on-social-media.aspx, February 2012.

[14] NYTimes. Facebook privacy: A bewildering tangle of options - graphic. http://www.nytimes.com/interactive/2010/05/12/business/facebook-privacy.html?ref=personaltech, December 2011.

[15] T. Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, December 2011.

[16] F. Press. Statistics — facebook. http://www.facebook.com/press/info.php?statistics, April 2012.

[17] SocioPatterns. Primary school – cumulative networks. http://www.sociopatterns.org/datasets/primary-school-cumulative-networks, April 2012.

[18] J. Travers and S. Milgram. An experiment study of the small world problem. *Sociometry*, Volume 32(4):425–443, 1969.

[19] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[20] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *2010 IEEE Symposium on Security and Privacy (SP)*, volume 1081-6011, pages 223–238. IEEE, May 2010.