A Study of ClickJacking Worm Propagation in Online Social Networks

Mohammad R. Faghani
Department of Computer Science and Engineering
York University
Toronto, Canada
faghani@cse.yorku.ca

Uyen T. Nguyen
Department of Computer Science and Engineering
York University
Toronto, Canada
utn@cse.yorku.ca

ABSTRACT

Clickjacking worms exploit online social network features such as Like and Share to propagate themselves in these networks. They trick a user into clicking on something different from what the user intends to click on. We discuss an implementation of such clickjacking malware and the way it propagates in a social network. We present simulation results that characterize the propagation of clickjacking malware in social networks. In particular, user habits of following posted links and the highly clustered structure of online social networks have significant impacts on the propagation speed of clickjacking malware.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Social network security, Malware propagation*; D.4.6 [**Operating Systems**]: Security and Protection—*Application Vulnerability*

General Terms

Security

Keywords

Malware, Online Social Networks, ClickJacking Worms

1. INTRODUCTION

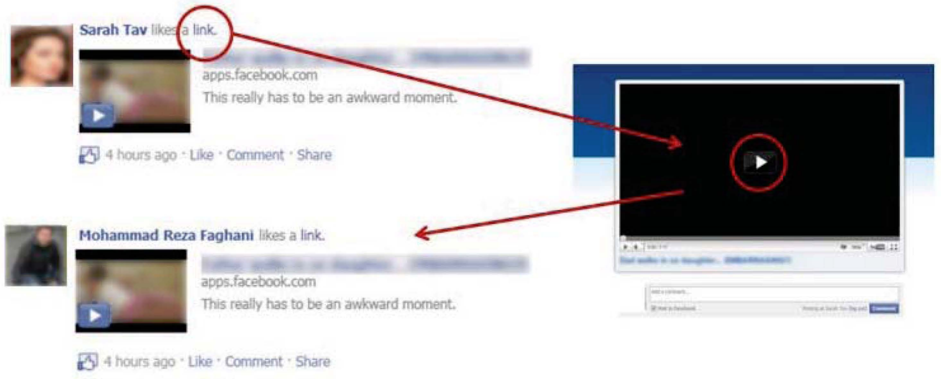
Online social networks such as Facebook, Twitter and MySpace have attracted hundreds of millions of people worldwide who use this service to connect and communicate with their friends, family and colleagues geographically distributed all around the world. This service cuts two ways, however. On one hand, OSNs are an ideal place for people to gather, communicate, socialize and share their common interests. On the other hand, malware creators often exploit the trust relationship among OSN users to propagate automated worms through online social networks for their own benefits. The first OSN worm, Samy, that hit MySpace in 2005 by exploiting a cross-site scripting (XSS) vulnerability in a MySpace web application infected about one million victims within 24 hours [1].

A clickjacking worm works by tricking a user into clicking on something he does not intend to do. For example, the user thinks he is clicking on a playback button to play a YouTube video clip while he is actually clicking on an invisible "Like" (or "Shared") button placed on top of the playback button. This action automatically posts a link to the spam site to the user's Facebook news feed, which will be shown to all his friends. The unintended "Like" action and the spam link will make the user's friends think that he has recommended the video, and many of them may follow the link to see the video. Figure 1 illustrates the process of a clickjacking malware propagation. The process continues until the malware is discovered by the OSN administrator and the spam links are removed, or the attacker stops the process himself (e.g, by removing the spam page containing the video).

The impacts of clickjacking worms can range from benign to harmful outcomes to victims. When a user unknowingly visits a web site, the attacker can make money through affiliated advertising programs. The more people "like" and subsequently visit the page, the more profit the attacker makes. A clickjacking worm can also trick users into enabling their webcams, invading their privacy [22]. In more serious attacks, clickjacking worms can redirect people to malicious web pages that host malware, which will be installed on the victims' computers using drive by download techniques.

There has not been any in-depth research on clickjacking worms and their propagation in *online social networks*. In fact, the topic of malware in OSNs has only been studied recently. However, existing works focus on prevention, detection, containment and elimination of malware [2, 3, 4, 5, 6], or on XSS worms [1] and Trojan worms [7]. Our work in this article focuses on characteristics of *clickjacking worm propagation in OSNs*, which will allow us to design more effective and resource-efficient countermeasures in the future.

We identify two major factors that have significant effects on the clickjacking worm propagation speed (infection rate) in an OSN. They are (1) user behaviors, namely, the probability of following a posted link and (2) the highly clustered structure of communities. We conducted simulations


Sarah Tav likes a link.

apps.facebook.com

This really has to be an awkward moment.

4 hours ago · Like · Comment · Share

Mohammad Reza Faghani likes a link.

apps.facebook.com

This really has to be an awkward moment.

4 hours ago · Like · Comment · Share

Figure 1: Self propagating process of clickjacking malware

on a real-world Facrbook subgraph to study the propagation characteristics of clickjacking malware in an OSN.

The remainder of this article is organized as follows. We discuss related work in Section 2 and describe an implementation of clickjacking worms for OSNs in Section 3. The simulation model and parameters are presented in Section 4. We analyze the simulation results in Section 5. We summarize the article and outline our future work in Section 6.

2. RELATED WORK

Among the first works on malware propagation in online social networks are [1, 7, 8]. Several other researchers also study malware propagation in OSNs via simulations [3, 9, 2]. However, they did not use graphs with the characteristics of OSNs (i.e., high clustering co-efficient, and low average shortest path distance) [3, 9], or limited their simulations to only a single network [2]. Yan et al. [2] describe three approaches for node monitoring in order to detect malware in OSNs using (1) node degree metric, (2) user activities and (3) network partition into small islands.

Defense and detection mechanisms against clickjacking have previously been studied [10, 11, 12, 13]. Rydstedt et al. [10] studied Alexa's top-500 frame busting technique − a technique which disallows a page to be framed − and showed that all of the implemented **clickjacking preventing techniques** can be circumvented. Niemietz et al. [11] discussed different clickjacking attack vectors, and introduced an automated detection system that is based on web page statistics. Johns and Lekies [12] proposed a likejacking[1] protection technique based on three pillars: Javascript visibility check, a secure in-browser communication protocol and integrity of essential DOM properties and APIs. Rehman et al. [13] proposed a browser-based solution to protect against cursor spoofing and clickjacking, which can help to detect likejacking in online social networks.

To the best of our knowledge, the work presented in this paper is the first that addresses the issue of clickjacking worm propagation in online social networks.

[1]Likejacking is a type of clickjacking that tricks users of a website into posting a "Like" update for the site unknowingly as discussed earlier.

3. CLICKJACKING WORMS: IMPLEMENTATION AND PROPAGATION

In this section, we discuss how a clickjacking malware can be implemenetd and propagated in an OSN. An attacker first creates fake profiles to infiltrate a social network. Using the fake profiles, they try to befriend as many real users as possible in order to spread a malware as widely and quickly as possible via these "friendships" and "Like" or "Shared" features.

The attacker then creates an enticing web page to lure people into viewing them. This web page, for example, may contain latest updates on breaking news, gossips on celebrities, exclusive video clips, or promotional items (e.g., coupons and free gift cards, which may or may not be given out). Lisitng 1 shows a code snippet in which the attacker embeds a YouTube video with a playback button (see Part 1).

In the second part, the attacker places a Facebook "Like" button on top of the playback button (see Part 2). For example, assume that the playback button is located at the coordinate (x, y) on the web page. The attacker will set the <fb> tag position in the style to place the "Like" button exactly at coordinate (x, y) (not shown here). To make the "Like" button invisible, its opacity is set to zero.

The attacker then clicks on the "Like" button, which posts a link to the spam site containing the video to the attacker's news feed. When his friends see the "Like" post, they will click on the link, which leads them to the video. When a user clicks on the playback button to view the video, she is actually clicking on the "Like" button. Her friends will see her (unintended) recommendation posted on her new feed and follow the same link. This process continues until the malware is detected and removed, or the attacker stops the propagation himself.

Sometimes, after the user clicks on the invisible button and realizes that no action is performed (e.g., the video is not played, or the next photo is not shown), they keep clicking on the button. To prevent users' frustration and suspicion (which eventually leads them to reporting the spam site to

Listing 1: Clickjacking worm code snippet

```
<--! Part1: Showing the video underneath the hidden
    like button -->

<iframe width="640" height="410" frameborder="0"
    allowfullscreen="" allowtransparency="true"
    src="Youtube.com/avideo.html" style="z-index:-1">


<--! Part2: Making the like button hide and put it on
    top of the play button -->

<fb:like id="fblike"
    href="currentsite.com/currentpage.html"
    style="opacity:0;filter:alpha(opacity=0);">
```

the network administrator or the anti-virus software), the attacker should write better code so that, after the first click, the invisible “Like” button will be removed to let the user click on the actual playback button. As a result, the user will see the video played and would think that the first click was not performed properly.

Facebook recently implemented some countermeasures to combat clickjacking worms. If the URL of a web page is deemed suspicious, Facebook will ask a user to confirm her “Like” action before a recommendation (and thus the spam link) is posted on the user’s news feed. This countermeasure can prevent a clickjacking malware from self propagating in some cases (e.g., well known malicious web sites that are black listed). Web sites or applications registered with Facebook are deemed legitimate and are not subject to “Like” action confirmation. Therefore, an attacker could register his application or web page to make it legitimate, and put the registration ID in the script in order to bypass the screening for “Like” confirmation. (Registering an application requires the attacker’s personal information such as name, phone number and mailing address. Stolen personal information can be bought cheaply on the black market.)

Moreover, clickjacking worms can propagate through other means outside social networks such as through email or online forums.

4. SIMULATION MODEL AND PARAMETERS

In this section, we review the characteristics of online social networks, and describe the network graph model and malware propagation model used in our simulations.

In several OSNs such as Facebook, LinkedIn, Orkut, and hi5, the relationship (friendship) between two users is mutual. Such an OSN can be represented by an undirected graph $G = (V, E)$ in which each vertex (or node) $v \in V$ represents a user, and an edge $e \in E$ between two vertices indicates the existence of a relationship (friendship) between the two respective users. In this article, we consider only OSNs that can be represented by undirected graphs. Our simulations were carried out on a real-world graph [14] that possess all the characteristics of a social network. The characteristics of online social networks, which are studied in [16], [17], [15] can be summarized as follows:

1. An OSN typically has a low average network distance, approximately equal to $\log(s)/\log(d)$, where s is the number of vertices (people), and d is the average vertex degree of the equivalent graph.

2. Online social networks typically show a high clustering property, or high local transitivity. That is, if person A knows B and C, then B and C are likely to know each other. Thus A, B and C form a friendship triangle. Let k denote the degree of a vertex v. Then the number of all possible triangles originated from vertex v is $k(k-1)/2$. Let f denote the number of friendship triangles of a vertex v in a social network graph. Then the clustering coefficient $C(v)$ of vertex v is defined as $C(v) = 2f/(k(k-1))$. The clustering coefficient of a graph is the average of the clustering coefficients of all of its vertices. In a real OSN, the average clustering coefficient is about 0.1 to 0.7.

3. Node degrees of a social network graph tend to be, or at least approximately, power-law distributed. The node degree of a power-law topology is a right-skewed distribution with a power-law Complementary Cumulative Density Function (CCDF) of $F(k) \propto k^{-\alpha}$, which is linear on a logarithmic scale. The power law distribution states that the probability for a node v to have a degree k is $P(k) \propto k^{-\alpha}$, where α is the power-law exponent [19].

For the simulations reported in this article, we used the Facebook social network graph constructed by McAuley and Leskovec [14]. The parameters and characteristics of this OSN graph are listed in Table 1. We also created an equivalent random graph (ERG) corresponding to the Facebook graph using the algorithm proposed by Viger and Latapy [18]. The random graph has the same node degree distribution as the equivalent Facebook graph. However, the other parameters may be different. For instance, an ERG usually has a lower clustering coefficient and network diameter than the original OSN graph. The parameters of an equivalent random graph generated based on the Facebook sub-graph are listed in Table 1.

Previous research has shown that a malware may propagate faster in an ERG than in the original OSN graph [8, 7]. An attacker may be able to obtain the graph of an OSN using a tool such as R [20] or Pajek [21]. He may then create ERGs based on the original OSN graph using an algorithm such as the one by Viger and Latapy [18]. We also study the propagation of clickjacking worms in ERGs to determine whether ERGs help or hinder the propagation of clickjacking worms in order to predict attack strategies.

We define an *event* or a *visit* in an OSN to be the action of visiting (accessing) a user’s home page or news feed. We assume that events in an OSN happen consecutively one after another. (Two different users may click on the same profile at the same time. Their access requests, however, will be queued at a server consecutively, waiting to be processed. The two events are thus considered to happen one after the other.)

Parameter	OSN	ERG
Number of vertices (people)	4,039	4,039
Number of edges	88,234	88,234
Average clustering coefficient	0.6055	0.06
Average shortest path length	3.692	2.59
Network diameter	8	5
Maximum node degree	1045	1045
Average node degree d	43.69	43.69
$\log(N)/\log(d)$	1.6	1.6

Table 1: The OSN and its Equivalent Random Graph

The simulation software is implemented using MATLAB. The simulation is of discrete-event type, consisting of discrete virtual time slots. A time slot is equivalent to an *event* defined above. In the first time slot (i.e., when the simulation starts, a user (node) is chosen randomly to be the attacker, who clicks the "Like" button on the spam site and the recommendation is posted on his news feed for all his friends to see. (Two users are friends if and only if their corresponding vertices in the OSN graph is connected by an edge $e \in E$.) In the next time slot, another user j is selected randomly with a probability of $1/N$ where N is total number of nodes in the network. If the user sees the spam link (i.e., one of her friends had "liked" the video/photo/page earlier), the user will follow the link with a probability α and get clickjacked (infected). (Some users are more cautious and do not click on just any link.) This process continue until the simulation is stopped.

Note that the spam link may be pushed down on a Facebook page by more recent activities. In this case, the user needs to scroll down the screen and checks for all new posts in order to see the spam link. Not all users have the habit of checking all their new post. We conducted an online survey involving 182 Facebook users from 18 countries around the world. 75% of the participants said that they would scroll all the posts to capture new posts. In our simulation we assume that if a spam link is posted on a user's wall, only 75% of these users actually see the spam link.

Each data point in the result graphs is the average of 100 runs, each with a different random seed. If a user's browser has add-on protections to prevent clickjacking scripts from running automatically, that user is considered not vulnerable to clickjack worms. We will consider only *vulnerable users* in our analysis and simulations.

5. SIMULATION RESULTS AND ANALYSIS

When the simulation started, a random user (node) was selected to be the attacker's fake profile. The attacker "Liked" a spam site and the recommendation was posted on his news feed. In the next time slot, a random user A was chosen with a probability of $1/N$ where N is total number of nodes in the network. If a spam link had previously posted on A's wall by a friend, A would actually scroll down to see the post with a probability $q = 0.75$ according to the statistics provided by our online survey. When A saw the spam link, she may or may not click on the link as some people are more cautious than others. Assume that a user would click on the spam link with a probability α and get infected[2]

The performance metric is the total number of infections (infected users) S as a function of the number of visits (time slots). Given the same number of visits, the smaller the value of S, the better. We carried out two sets of experiments, one using the Facebook sub-graph and the other using an random equivalent graph (ERG) as described in the above section.

Experiment 1. Using the Facebook sub-graph network, we varied the probability p of getting infected from 0.25 to 0.75. The graph in Figure 2 shows the total number of infections as a function of the number of visits for $p = 0.25, 0.5, 0.75$. The results show that the higher the probability p, the more users are infected given the same number of visits. For instance, after the $15,000^{th}$ visit, the total number of infected users is 2,975 for $p = 0.75$ while that number is only 756 for $p = 0.25$.

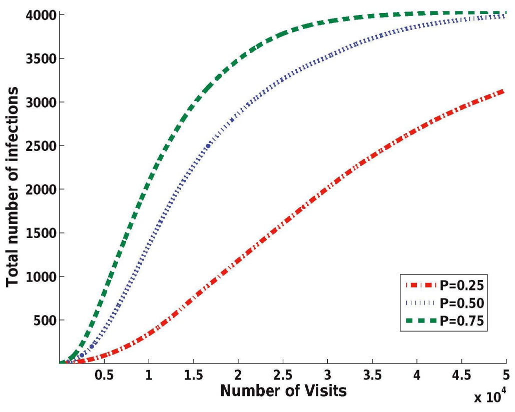


Figure 2: Clickjacking worm proapgation for different values of p in the OSN

Using the same data collected in this experiment, we plotted a graph that shows the impact of probability p on the malware propagation. In particular, the graph in Figure 3 shows the number of visits required to infect 90% of the population (3,635 out of 4,039 users) as a function of p. As the value of p increases, the number of visits required to infect 90% of the population goes down significantly. For instance, this number is 49,538 for $p = 0.3$ and only 18,780 for $p = 0.8$. That is, the more cautious users are about clicking on unknown links, the longer it takes a malware to infect the same number of profiles.

If we assume that people visit the OSN based on a Poisson process, we can convert the number of visits into an actual time scale (hours). In the online survey mentioned earlier, 92.2% of the participants said that they visit a social network at least once a day. Given the OSN used in this experiment (4,039 users), this results in 3,724 people visiting the website

[2]In this paper, we assume that all users have the same probability α. In future work, we will consider the case in which different users have different probabilities of following an unknown link. The probability p for a user to be infected is thus $p = q \times \alpha$.

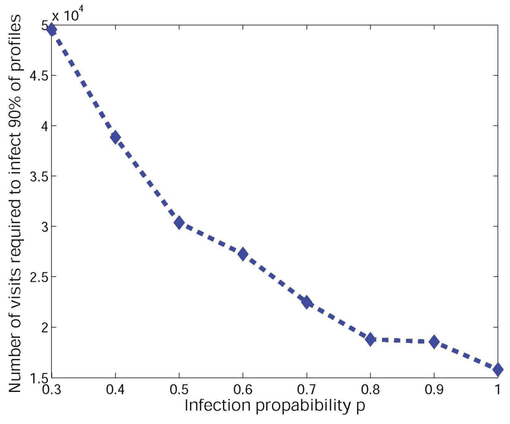


Figure 3: Total number of visits required to infect 90% of the population

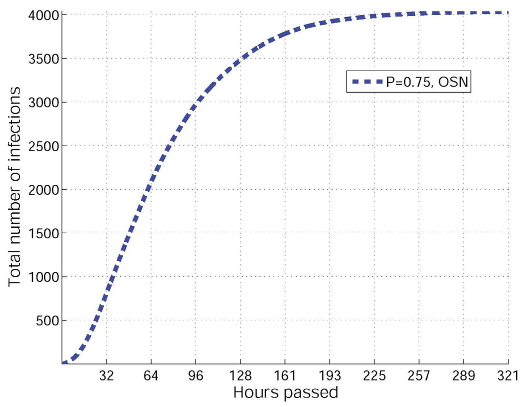


Figure 4: Total number of infections as a function of number of hours ($p = 0.75$)

at least once a day. This means an average of 155 visits per hour. Therefore, for a unit of one hour, we have a Poisson process with $\lambda = 155$. Since we assume that the average number of visits per hour is $\lambda = 155$, the inter-arrival time of the users follows an exponential distribution with $\lambda^{-1} = 155$. If we map the x-axis of the graph in Figure 2 to the time scale based on the above calculation, we obtain the graph in Figure 4 for the case where $p = 0.75$. The graph shows that the malware can infect half of the population in roughly two and a half days (63 hours), and the whole network in 10 days.

Experiment 2. We repeated Experiment 1 using the equivalent random graph whose parameters are listed in Table 1. The results are shown in Figure 5. As in the previous case, the higher the probability p, the more infections observed in the network. For example, after the $15,000^{th}$ visit, the total number of infections is 3,897 for $p = 0.75$ and 1,978 for $p = 0.25$.

To compare the propagation speed of the malware in the original OSN and the ERG, we transfered the curves from Figure 2 and Figure 5 for $p = 0.75$ to Figure 6. The com-

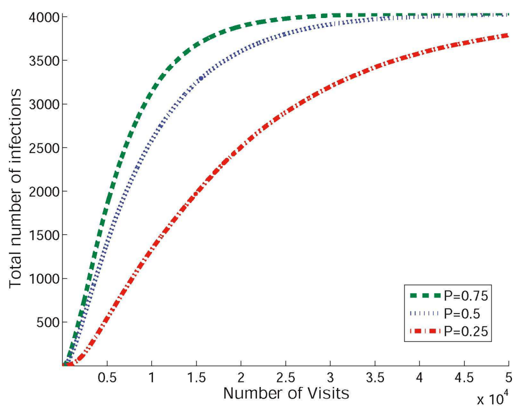


Figure 5: Clickjacking worm proapgation for different values of p in the ERG

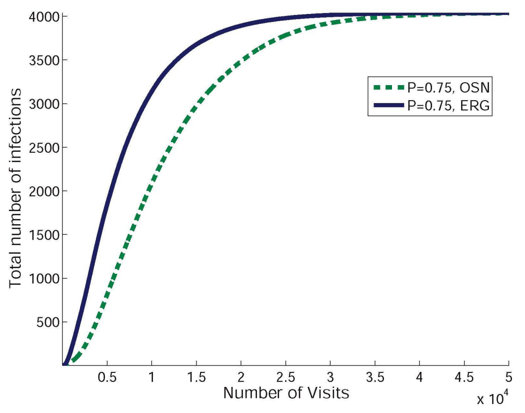


Figure 6: Comparing ERG vs. OSN for $p = 0.75$

bined graph shows that the malware propagates faster in the ERG network than in the original OSN. For example, after the $15,000^{th}$ visit, the total number of infected profiles is 2,975 in the original OSN, while this number is 3,897 in the ERG network. We came to the same conclusion when comparing the number of infections in the original OSN to that in the ERG network for other p values. That is, the ERG network enables a malware to spread faster than a real OSN. The reason is that the ERG has a lower clustering coefficient than the original OSN graph, 0.06 vs. 0.6. A higher clustering coefficient implies that a message will circulate for a while in a community among friends before reaching to other parts of the OSN, slowing down the malware propagation.

6. CONCLUSION

We discuss an implementation of clickjacking worms that exploit social network features such as Like and Share to propagate themselves. We present simulations results that demonstrate that user behaviours have an impact on the propagation of clickjacking malware: the more cautious users are about unknown links, the more slowly clickjacking malware propagates. Furthermore, the high clustering structure of social networks helps to slow down the propagation of such malware. In our future work, we will investigate countermeasures against clickjacking worms in OSNs as well as resource-efficient detection mechanisms.

7. REFERENCES

[1] Mohammad R. Faghani and Uyen Tran Nguyen. A Study of XSS Worm Propagation and Detection Mechanisms in Online Social Networks. *IEEE Transactions on Information Forensics and Security*, Volume 8, Issue 11, 2013.

[2] Guanhua Yan, Guanling Chen, Stephan Eidenbenz, and Nan Li. 2011. Malware propagation in online social networks: nature, dynamics, and defense implications. *In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (ASIACCS '11). ACM, New York, NY, USA, 196-206.

[3] Wei Xu, Fangfang Zhang, and Sencun Zhu. 2010. Toward worm detection in online social networks. *In Proceedings of the 26th Annual Computer Security Applications Conference* (ACSAC '10). ACM, New York, NY, USA, 11-20.

[4] Tao Stein, Erdong Chen, and Karan Mangla. 2011. Facebook immune system. *In Proceedings of the 4th Workshop on Social Network Systems* (SNS '11). ACM, New York, NY, USA, , Article 8 , 8 pages.

[5] Nam P. Nguyen, Ying Xuan, My T. Thai. A novel method for worm containment on dynamic social networks. *In Proceedings of the Military Communcation Conference*, San Jose, CA, USA, pp. 2180-2185, 2010.

[6] Yinzhi Cao, Vinod Yegneswaran, Phillip Porras, and Yan Chen. PathCutter: Severing the Self-Propagation Path of XSS JavaScript Worms in Social Web Networks. *In Proceedings of 19th Symposium on Network Distributed Systems*, San Diego, CA, USA, 2012

[7] Mohammad R. Faghani, Ashraf, Chung-Horng Lung. A study of trojan propagation in online social networks. *In Proceedings of 5th International Conference of IEEE IFIP New Technology Security*, Istanbul, Turkey, pp. 1-5, 2012.

[8] Mohammad R. Faghani and Hossein Saidi. Malware Propagation in Online Social Networks. *In Proceedings of Internation Conference on Malicious and Unwanted Programs (Malware09)*, Montreal, Canada, 2009.

[9] Meytal Tubi, Rami Puzis, and Yuval Elovici. Deployment of DNIDS in Social Networks. *Proceedings of IEEE Intelligence and Security Informatics Conference*, New Brunswick, NJ, USA, pp. 59-65, 2007.

[10] Gustav Rydstedt, Elie Bursztein, Dan Boneh and Collin Jackson. Busting frame busting: a study of clickjacking vulnerabilities at popular sites. *In Proceedings of IEEE Oakland Web 2.0 Security and Privacy Workshop*. 2010

[11] Marcus Niemietz. UI Redressing: Attacks and Countermeasures Revisited. *In Proceedings of CONFidence*. 2011

[12] Martin Johns, Sebastian Lekies. Tamper-Resistant LikeJacking Protection. *Lecture Notes in Computer Science, Research in Attacks, Intrusions, and Defenses.* Springer, pp. 265-285. 2013

[13] Ubaid Ur Rehman, Waqas Ahmad Khan, Nazar Abbas Saqib and Muhammad Kaleem. On Detection and Prevention of Clickjacking Attack for OSNs. *In Proceedings of Frontiers of Information Technology (FIT)*. 2013.

[14] Julian Mcauley and Jure Leskovec. Learning to Discover Social Circles in Ego Networks. *In Proceedings of Neural Information Processing Systems*. 2012.

[15] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. 2007. Analysis of topological characteristics of huge online social networking services. *In Proceedings of the 16th international conference on World Wide Web (WWW '07)*. ACM, New York, NY, USA, 835-844.

[16] A. Dekker. Realistic Social Networks for Simulation using Network Rewiring. In: Proceedings of International Congress on Modelling and Simulation, 2008.

[17] Peter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E 65*. pp. 026107-1:4. 2002.

[18] Fabien Viger and Matthieu Latapy. 2005. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *In Proceedings of the 11th annual international conference on Computing and Combinatorics (COCOON'05)*, Lusheng Wang (Ed.). Springer-Verlag, Berlin, Heidelberg, 440-449.

[19] M. J. Newman. Power laws, Pareto distributions and Zipf law. *Contemporary Physics*. vol. 46(5). pp. 323-351. 2004.

[20] Ross Ihaka and Robert Gentleman. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics*. vol. 5(3). pp. 299-314. 1996.

[21] Vladimir Batagelj and Andrej Mrvar. Pajek a Program for large network analysis. *Connections*. 21: 47-57. 1998.

[22] Robert Hansen and Jeremiah Grossman. Clickjacking. SecTheory. Available from: http://www.sectheory.com/clickjacking.htm