DOI:10.1145/3387720

Measuring and Mitigating OAuth Access Token Abuse by Collusion Networks

By Shehroze Farooqi, Fareed Zaffar, Nektarios Leontiadis, and Zubair Shafiq

Abstract

We uncovered a thriving ecosystem of large-scale reputation manipulation services on Facebook that leverage the principle of collusion. *Collusion networks* collect OAuth access tokens from colluding members and abuse them to provide fake likes or comments to their members. We carried out a comprehensive measurement study to understand how these collusion networks exploited popular third-party Facebook applications with weak security settings to retrieve OAuth access tokens. We infiltrated popular collusion networks using honeypots and identified more than one million colluding Facebook accounts by "milking" these collusion networks. We disclosed our findings to Facebook and collaborated with them to implement a series of countermeasures that mitigated OAuth access token abuse without sacrificing application platform usability for third-party developers.

1. INTRODUCTION

Reputation is a fundamental tenet of online social networks. People trust the information that is posted by a reputable social media account or is endorsed (e.g., liked) by a large number of accounts. Unfortunately, reputation fraud is prevalent in online social networks. A number of black-hat reputation manipulation services target popular online social networks.[13,19] To conduct reputation manipulation, fraudsters purchase fake accounts in bulk from underground marketplaces,[21] use infected accounts compromised by malware,[18] or recruit users to join collusion networks.[22]

Online social networks try to counter reputation manipulation activities on their platforms by suspending suspicious accounts. Prior research on detecting reputation manipulation activities in online social networks can be broadly divided into two categories: *(a)* identifying temporally synchronized manipulative activity patterns[12,16]; *(b)* identifying individual accounts suspected to be involved in manipulative activity based on their social graph characteristics.[11,25] Recent studies have shown that fraudsters can circumvent these detection methods by incorporating "normal" behavior in their activity patterns.[13,23] Defending against fraudulent reputation manipulation is an ongoing arms race between fraudsters and social network operators.[3,8]

In this paper, we uncovered a thriving ecosystem of reputation manipulation services on Facebook that leverage the principle of *collusion*. In these collusion networks, members like other members' posts and in return receive likes on their own posts. Such collusion networks of significant size enable members to receive a large number of likes from other members, making them appear much more popular than they actually are. As expected, colluding accounts are hard to detect because they mix real and fake activity. Our goal in this paper is to understand their methods of coordination and execution to develop effective and long-lasting countermeasures.

OAuth Access Token Leakage. To understand the extent of the problem collusion networks pose, we analyzed popular Facebook collusion networks. We found that collusion networks conduct reputation manipulation activities by exploiting popular third-party Facebook applications with weak security settings. Third-party Facebook applications gain restricted access to users' accounts using OAuth 2.0,[14] which is an authorization framework. When a user authenticates an application using OAuth 2.0, an *access token* is generated. Collusion networks collect these OAuth access tokens for applications, which utilize the *implicit* mode in OAuth 2.0, with help from colluding members. These access tokens are then used to conduct activities on behalf of these applications and colluding accounts. Using a large pool of access tokens, collusion networks provide likes and comments to their members on an on-demand basis. We found that popular collusion networks exploited a few popular Facebook applications. However, our analysis of top 100 Facebook applications revealed that more than half of them are susceptible to access token leakage and abuse by collusion networks. Although prior research has reported several security weaknesses in OAuth and its implementations,[6,15,20] we are the first to report large-scale OAuth access token leakage and abuse. As OAuth 2.0 is also used by many other large service providers, their implementation may also be susceptible to similar access token leakage and abuse.

Milking Collusion Networks Using Honeypots. We deployed honeypots to conduct a large-scale measurement study of popular Facebook collusion networks. Specifically, we created honeypot Facebook accounts, joined collusion networks, and "milked" them by requesting likes and comments on posts of our honeypot accounts. We then monitored and analyzed our honeypots to understand the strategies used by collusion networks to manipulate reputation. We identified more

The original version of this paper was published in *Proceedings of the Internet Measurement Conference, ACM, 2017*; https://conferences.sigcomm.org/imc/2017/papers/imc17-final235.pdf

than one million unique colluding accounts by milking collusion networks. As part of the milking process, we submitted more than 11K posts to collusion networks and received a total of more than 2.7 million likes. We identified the membership size of collusion networks by tracking the number of unique accounts that liked the posts of our honeypot accounts. Our membership estimate of these collusion networks is up to 295K for hublaa.me followed by 233K for official-liker.net in the second place. The short URLs used by collusion networks to retrieve access tokens have more than 289 million clicks to date. Our analysis of short URLs shows that popular collusion networks are used daily by hundreds of thousands of members. Collusion networks monetize their services by displaying advertisements on their heavily visited websites and offering premium reputation manipulation plans.

Countermeasures. We disclosed our findings to Facebook and worked with them to mitigate these collusion-based reputation manipulation services. Although we identified a wide range of possible countermeasures, we decided to implement the countermeasures that provide a suitable tradeoff between detection of access token abuse and application platform usability for third-party developers. For instance, we do not block the third-party applications exploited by collusion networks because it will negatively impact their millions of legitimate users. We do not disallow OAuth implicit mode, which is optimized for browser-based applications, because it will burden third-party developers with prohibitive costs associated with server-side application management. As part of countermeasures, we first introduced rate limits to mitigate access token abuse but collusion networks quickly adapted their activities to avoid these rate limits. We then started invalidating access tokens that are milked as part of our honeypot experiments to mitigate access token abuse by collusion networks. We further rate limited and blacklisted the IP addresses and autonomous systems (ASes) used by collusion networks to completely cease their operations.

2. OAUTH ACCESS TOKEN ABUSE

In this section, we first provide a background of Facebook's third-party application ecosystem and then discuss how attackers can exploit these applications to abuse their OAuth access tokens.

2.1. Background

All major online social networks provide social integration APIs. These APIs are used for third-party application development such as games, entertainment, education, utilities, etc. These applications acquire read/write permissions from the social network to implement their functionalities. Popular social network applications have tens of millions of active users and routinely conduct read/write operations on behalf of their users.

Facebook also provides a development platform for third-party applications. Facebook implements OAuth 2.0 authorization framework[14] which allows third-party applications to gain restricted access to users' accounts without sharing authentication credentials (i.e., username and password). When a user authenticates an application using OAuth 2.0, an *access token* is generated. This access token is an opaque string that uniquely identifies a user and represents a specific *permission scope* granted to the application to perform read/write actions on behalf of the user. A permission scope is a set of permissions requested by the application to perform actions on behalf of the user.

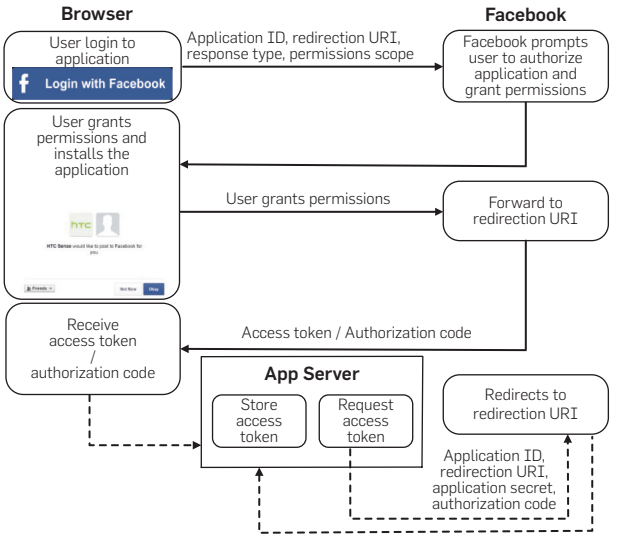
There are two types of permissions that an application may request. The first type of basic permissions does not require Facebook's approval. They include access to profile information, email addresses, and friend lists. The second type of sensitive permissions (e.g., `publish_actions`) requires Facebook's approval.[4] These permissions allow third-party applications to conduct certain actions on behalf of a user, e.g., posting status updates, generating likes and comments.

Access tokens are invalidated after a fixed *expiration duration*. They can be categorized as short-term or long-term based on their expiration duration. Facebook issues short-term access tokens with 1–2 hours expiration duration and long-term access tokens with approximately 2 months expiration duration.

OAuth 2.0[14] provides two workflows to generate an access token: client-side flow (also referred to as *implicit mode*) and server-side flow (also referred to as *authorization code mode*).[a] Both workflows are similar with few changes in request parameters and some additional steps in the server-side flow. Figure 1 illustrates the OAuth 2.0 workflow of a Facebook application to generate an access token for client-side and server-side authorizations.

[a] In addition to implicit and authorization code modes, OAuth 2.0 supports *resource owner password credentials mode* and *client credentials mode*. The former mode is used by clients to give their credentials (username and password) directly to the applications. The latter mode does not involve any client interaction and is used by applications to access their resources. We do not discuss these modes because they are not used to generate user access tokens.

Figure 1. Workflow of Facebook applications.



- The flow is initiated by directing a user to Facebook's authorization server by clicking on a login button. The request to the authorization server includes application ID, redirection URI, response type, and a permission scope. The application ID is a unique identifier assigned to every Facebook application. The redirection URI is configured in the application settings. The response type is set as "token" to return access token in a client-side flow and is set as "code" to return an authorization code in a server-side flow.
- Facebook's authorization server validates the request and prompts the user to authorize the application and grant permissions in the browser. User authorizes the application and grants the requested permissions to the application.
- Facebook redirects the user to the redirection URI along with an access token or an authorization code in the URL fragment. For the client-side flow, an access token is returned in response which is retrieved and stored by the application terminating the client-side flow. For the server-side flow, an authorization code is returned in response and the following additional step is required.
- The authorization code is exchanged for an access token by requesting Facebook's authorization server through the application's server.[5] The request includes application ID, redirection URI, authorization code, and application secret. The request to exchange an authorization code for an access token is authenticated using the application secret.

The access tokens are then used by applications to perform the Facebook Graph API requests on behalf of users. For each request, an application is generally required to pass on application ID, application secret, and the corresponding access token. As we discuss next, the application secret may not be mandatory to make these requests.

2.2. Identifying susceptible applications

Applications select a suitable OAuth flow based on their access token usage scenarios. Server-side flows are by design more secure than client-side flows because they do not expose access tokens at the browser. Facebook provides an option to disable client-side flow from application settings. Facebook recommends third-party applications to disable client-side flow if it is not used.[4] The client-side flow is typically allowed by applications that make Facebook Graph API calls only from the client side. For example, the client-side flow is used by browser-based applications which cannot include application secret in client-side code. In fact, some client-side applications may not have an application server at all and perform Graph API requests only from the browser using JavaScript. If the application secret is required, applications will have to expose their application secret in the client-side flow. It is noteworthy that the application secret is treated like a password and hence it should not be embedded in the client-side code.

Prior work has shown that attackers can retrieve access tokens by exploiting security weaknesses in the OAuth protocol and its implementations.[6, 15, 20] Facebook applications that use client-side flow and do not require application secret are susceptible to access token leakage and abuse. For example, attackers can retrieve access tokens in client-side flows by eavesdropping,[20] cross-site scripting,[17, 20] or social engineering techniques.[10] A leaked access token has serious security and privacy repercussions depending on its authorized resources. Attackers can abuse leaked access tokens to retrieve users' personal information. Attackers can also abuse leaked access tokens to conduct malicious activities such as spreading spam/malware.

We implemented a Facebook application scanning tool to identify applications that are susceptible to access token leakage and abuse. Our tool uses Selenium and Facebook SDK for Python to launch the application's login URL and install the application on a test Facebook account with the full set of permissions. We first infer the OAuth redirection URI used by the application by monitoring redirections during the Facebook login flow. Using the OAuth redirection URI, we install the application on the test Facebook account with the permissions that were originally acquired by the application. If the application is successfully installed, we retrieve the access token at the client-side from the application's login URL. Using the access token, we make an API call to retrieve the public profile information of the test Facebook account and like a test Facebook post. If we are able to successfully conduct these operations, we conclude that the application can be exploited for reputation manipulation using leaked access tokens.

We analyzed top 100 third-party Facebook applications using our scanning tool. Our tool identified 55 susceptible applications, out of which 46 applications were issued short-term access tokens and 9 applications were issued long-term access tokens. Short-term access tokens pose a limited threat because they are required to be refreshed after every 1–2 hours. On the other hand, long-term access tokens provide a 2-month-long time window for an attacker. The highest ranked susceptible application which was issued long-term access tokens had about 50 million monthly active users. In fact, many of these susceptible applications had millions of monthly active users, which can cloak access token abuse by attackers.

3. COLLUSION NETWORKS

A number of reputation manipulation services provide likes and comments to Facebook users based on the principle of collusion: members like other members' posts, and in return receive likes from other members. As discussed earlier, these collusion networks exploit Facebook applications with weak security settings. Collusion networks of significant size can enable members to escalate their reputation, making them appear much more popular than they actually are.

We first surveyed the landscape of Facebook collusion networks by querying search engines for the relevant keywords, such as "Facebook AutoLiker," "Status Liker," and "Page Liker," that were found on a few well-known collusion network websites. We compiled a list of 50 such

[b] https://blog.alexa.com/marketing-research/alexa-rank/

websites and used Alexa Rank,[b] which is a measure of website popularity, to shortlist popular collusion networks. It is noteworthy that the top 8 collusion networks were ranked within the top 100K. For example, hublaa.me was ranked around 8K and 18% of their visitors were from India, where it was ranked within the top 3K sites. It is interesting to note that other collusion networks also got most of their traffic from countries such as India, Egypt, Turkey, and Vietnam.

We investigated popular collusion networks to understand the features they offer and to identify Facebook applications that they exploited. Collusion networks ask users to install a Facebook application and submit the generated access token in a textbox on their website. The installation link redirects users to a Facebook dialog mentioning the application's name. Table 1 lists the applications used by popular collusion networks along with their statistics retrieved from the Facebook Graph API. Using our tool, we verified that these Facebook applications used client-side flow and did not require application secret for making the Graph API calls. We observed that *HTC Sense*, which is used by several popular collusion networks, was ranked at 40 and had on the order of a million daily active users (DAU). *Nokia Account* was ranked at 249 and had approximately a hundred thousand daily active users. Similarly, *Sony Xperia smartphone* was ranked at 886 and had on the order of 10,000 daily active users. It is noteworthy that collusion networks cannot create and use their own applications because they would not pass Facebook's strict manual review process for applications that require write permissions.[1] However, collusion networks can (and do sometimes) switch between existing legitimate applications that are susceptible to access token leakage and abuse.

Most collusion networks have a similar web interface and they all provide a fairly similar user experience. Figure 2 illustrates the workflow of Facebook collusion networks.

- A user visits the collusion network's website and clicks on the button to install the application. The website redirects the user to the application authorization dialog URL. The user is asked to grant the requested permissions and install the application.
- The user returns to the collusion network website after installing the application and clicks on the button to retrieve the access token. The website again redirects the user to the Facebook authorization dialog URL with *view-source* appended. The authorization dialog redirects the user to a page that contains the access token as a query string in the URL. The use of *view-source* stops the authorization dialog from further redirections. The user manually copies the access token from the address bar and submits it at a textbox on the collusion network website.
- The collusion network saves the access token and redirects the user to an admin panel, where the user can request likes and comments. Some collusion networks require users to solve CAPTCHAs and/or make users watch ads before allowing them to request likes and comments.

Table 1. Facebook applications used by popular collusion networks.

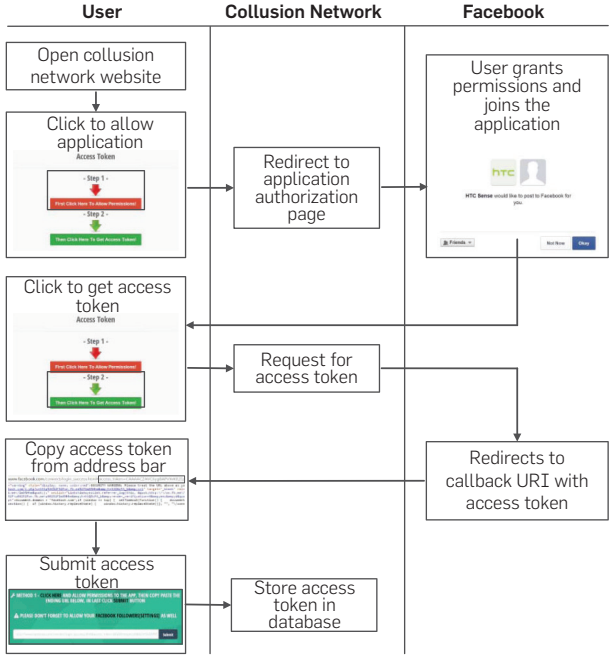
Application identifier	Application name	DAU	DAU rank	MAU	MAU rank
41158896424	HTC Sense	1M	40	1M	85
200758583311692	Nokia Account	100K	249	1M	213
104018109673165	Sony Xperia	10K	866	100K	1563

4. MEASURING COLLUSION NETWORKS

Honeypots have proven to be an effective tool to study reputation manipulation in online social networks.[13,24] The basic principle of honeypots is to bait and deceive fraudsters for surveilling their activities. In order to investigate the operation and scale of Facebook collusion networks, we deployed honeypots to "milk" them.

We created new Facebook honeypot accounts and joined different collusion networks using the workflow described in Section 3. Our honeypot accounts regularly posted status updates and requested collusion networks to provide likes/comments on these posts. Soon after the submission of our requests to collusion networks, we noticed sudden bursts of likes and comments by a large number of Facebook accounts which were part of the collusion network. As repeated requests result in likes/comments from many unique Facebook accounts, we can uncover the memberships of collusion networks by making a large number of reputation manipulation requests. Our goal is to estimate the scale of collusion networks by tracking their member Facebook accounts. We also want to understand the tactics used by collusion networks to stay under the radar and avoid detection.

Figure 2. Workflow of Facebook collusion networks.



4.1. Experimental design

We registered 22 new Facebook accounts intended to be used as active honeypots for studying popular collusion networks. Each honeypot account joined a different collusion network. In an effort to actively engage collusion networks, our honeypot accounts regularly posted status updates on their timelines and requested the collusion networks to provide likes/comments on them. It was challenging to fully automate this process because collusion networks employ several tactics to avoid automation. For example, some collusion networks impose fixed or random delays between two successive requests. Many collusion networks redirect users through various redirection services before allowing request submission. Several collusion networks require users to solve a CAPTCHA in order to login and before making each request. To fully automate our honeypots, we used a CAPTCHA solving service[2] for automatically solving CAPTCHAs and Selenium for submitting requests to collusion networks. We continuously posted status updates and requested collusion networks to provide likes/comments over the duration of approximately 3 months from November 2015 to February 2016.

4.2. Data collection

We regularly crawled the timelines of our honeypot Facebook accounts to log incoming likes and comments provided by collusion networks. The number of unique Facebook accounts who liked or commented on a honeypot account is an estimate of the collusion network's size. Note that our membership estimate is strictly a lower bound because we may not have observed all collusion network accounts, which are randomly picked from a large pool of access tokens. We also crawled the activity logs of our honeypot accounts to collect outgoing likes and comments.

4.3. Size of collusion networks

Milking collusion networks. We posted status updates from our honeypot accounts and requested collusion networks to provide likes on the posts. Figure 3 plots the cumulative distribution of likes and unique accounts milked by our honeypots for a collusion network that represents the behavior of most of the collusion networks. We observed that the count of new unique accounts steadily declined even though the new like count remains constant. The decline represents diminishing returns due to the increased repetition in users who liked the posts of our honeypot accounts. Specifically, due to the random sampling of users from the database of access tokens, the likelihood of user repetition increases as we post more status updates for a honeypot account. It is important that we milk collusion networks as much as possible to accurately estimate their membership size. Although we were able to max out many collusion networks, we faced some issues for a few collusion networks. For example, djliker.com and monkeyliker.com imposed a daily limit of 10 requests, thus we were not able to fully max out these collusion networks. Moreover, arabfblike.com and a few other collusion networks suffered from intermittent outages when they failed to respond to our requests for likes. The set of unique accounts who liked posts of our honeypot accounts were collusion network members. Table 2 shows that the membership size of collusion networks varied between 295K for hublaa.me to 834 for fast-liker.com. We note that hublaa.me had the largest membership size at 295K accounts, followed by official-liker.net at 233K and mg-likers.com at 178K. The membership size of all collusion networks in our study summed up to 1,150,782. As we discuss later, some accounts were part of multiple collusion networks. After eliminating these duplicates, the total

Figure 3. Cumulative distribution of likes and unique accounts.

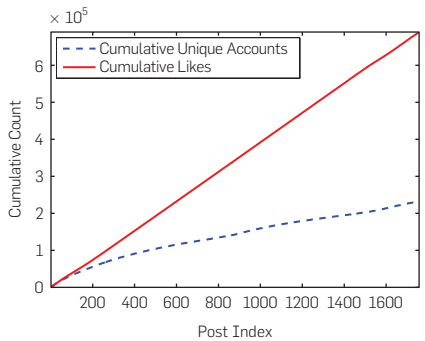


Table 2. Statistics of the collected data for all collusion networks.

Collusion network	Incoming activities: Total number of posts	Incoming activities: Total number of likes	Number of activities	Outgoing activities: Number of target accounts	Outgoing activities: Membership size
hublaa.me	1421	496,714	145	46	294,949
official-liker.net	1757	685,888	1955	846	233,161
mg-likers.com	1537	379,475	1524	911	177,665
monkey-liker.com	710	165,479	956	356	137,048
f8-autoliker.com	1311	331,923	2542	1254	72,157
djliker.com	471	70,046	360	316	61,450
autolikes-groups.com	774	202,373	1857	885	41,015
4liker.com	269	71,059	2254	1211	23,110
myliker.com	320	32,821	1727	983	18,514
kdliker.com	599	82,736	1444	626	18,421
oneliker.com	334	24,374	956	483	18,013
fb-auto-likers.com	244	19,552	621	397	16,234
autolike.vn	139	35,425	2822	1382	14,892
monsterlikes.com	495	72,755	2107	671	5168
postlikers.com	96	8613	2590	1543	4656
facebook-autoliker.com	132	4461	2403	1757	3108
realliker.com	105	19,673	2362	846	2860
autolikesub.com	286	25422	1531	717	2379
kingliker.com	107	5072	1245	587	2243
rockliker.net	99	4376	82	39	1480
arabfblike.com	311	4548	68	31	1328
fast-liker.com	232	10,270	1472	572	834
All	11,751	2,753,153	33,023	16,459	1,150,782

number of unique accounts across all collusion networks was 1,008,021.

4.4. Collusion network activities

Incoming activities. Table 2 summarizes the statistics of the data collected for different collusion networks using our honeypot accounts. In total, we submitted more than 11K posts to collusion networks and garnered more than 2.7 million likes. As shown in Figure 3, we observe that status updates typically received a fixed number of likes per request, ranging between 14 and 390 across different collusion networks. For example, official-liker.net, f8-autoliker.com, and myliker.com provided approximately 400, 250, and 100 likes per request, respectively.

Outgoing activities. Collusion networks also used our honeypot accounts to conduct reputation manipulation activities on other Facebook accounts and pages. In total, our honeypot accounts were used by collusion networks to like more than 33K posts of 16K accounts. We observed that some collusion networks used our honeypots more frequently than others. For example, autolike.vn used our honeypot accounts to provide a maximum of 2.8K likes on posts of 1.3K accounts.

5. COUNTERMEASURES

Ethical considerations. Before conducting any experiments, we received a formal review from our local Institutional Review Board (IRB) because we collected some publicly available account information such as posts and likes. We enforced several mechanisms to protect user privacy. For example, we did not store any personally identifiable information. We were aware that our honeypot accounts were used by collusion networks to conduct some reputation manipulation activities. We argue that these activities represented a small fraction of the overall reputation manipulation activities of collusion networks. Thus, we do not expect normal user activity to be significantly impacted by our honeypot experiments. The benefit of our honeypot approach in detecting collusion network accounts far outweighs the potential harm to regular Facebook users. To further minimize harm, as discussed next, we disclosed our findings to Facebook to remove all artifacts of reputation manipulation during our measurements as well as investigate countermeasures to mitigate collusion network activities.

Before implementing any countermeasures in collaboration with Facebook, we performed honeypot experiments for approximately 10 days to establish a baseline of collusion network activities. We repeated the honeypot milking experiments for popular collusion networks starting August 2016 (and continued until mid-October 2016). Figure 4 shows the average number of likes received by our honeypots for two popular collusion networks. We do not show results for other collusion networks due to space constraints. As we discuss next, while we considered a wide range of countermeasures, we decided to implement countermeasures that provide a suitable tradeoff between detection of access token abuse and application platform usability for third-party developers.

We observed that collusion networks exploited a few applications (listed in Table 1) to conduct reputation manipulation activities. Therefore, we can immediately disrupt all collusion networks by suspending these applications. As collusion networks can switch between several other susceptible applications, we would need to suspend them as well. Suspending these applications is a relatively simple countermeasure to implement; however, it will negatively impact their millions of legitimate users. We can also make changes in Facebook's application workflow to stop access token abuse by collusion networks. For example, we can mandate application secret (thereby forcing server-side operations) for liking/commenting activities that require `publish_actions` permissions.[4,7] As a result of this restriction, collusion networks will not be able to conduct reputation manipulation activities even if they retrieve access tokens from colluding users. However, many Facebook applications solely rely on client-side operations for cross-platform interoperability and to reduce third-party developer costs of server-side application management.[4,14] Therefore, mandating application secret would adversely impact legitimate use cases for these Facebook applications.

5.1. Access token rate limits

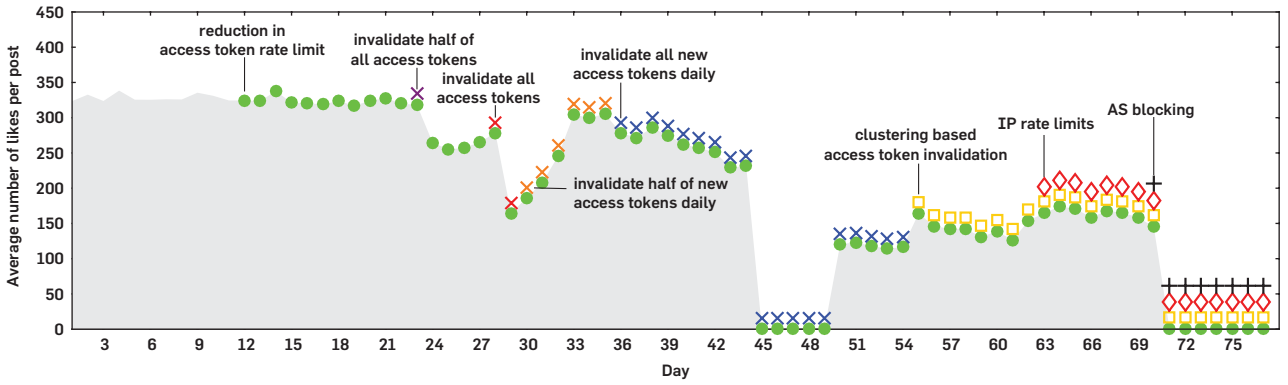
As the first countermeasure, we imposed restrictions on access tokens to mitigate abuse by collusion networks. Facebook employs rate limits to restrict excessive activities performed by an access to ken. As collusion network activities slip under the current rate limit, we reduced the rate limit by more than an order of magnitude on day 12 as marked by green circles in Figure 4. We observed a sharp initial decrease in activities for official-liker.net. Specifically, the average number of likes provided by official-liker.net decreased from more than 400 to less than 200 on day 16. However, official-liker.net started to bounce back after approximately 1 week. Moreover, this countermeasure did not impact hublaa.me. We surmise that both of these collusion networks had a large pool of access tokens which limited the need to repeatedly use them. Therefore, these collusion networks were able to stay under the reduced access token rate limit while maintaining their high activity levels. We did not reduce the rate limit further to avoid potential false positives.

5.2. Honeypot-based access token invalidation

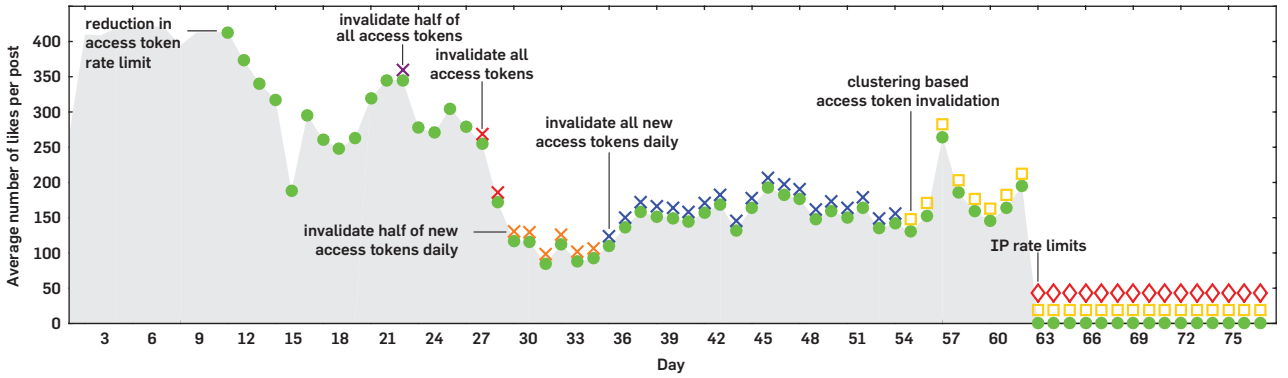
We next invalidated access tokens of colluding accounts which were identified as part of our honeypot experiments. In the first 22 days, we milked access tokens of 283K and 41K users for hublaa.me and official-liker.net, respectively. We expect that invalidation of these access tokens will curb collusion network activities. To this end, we invalidated randomly sampled 50% of the milked access tokens on day 23 as marked by a black cross in Figure 4. We observed a sharp decrease in collusion network activities. Specifically, the average number of likes provided by hublaa.me decreased from 320 to 250 and for official-liker.net decreased from 350 to 275. Unfortunately, this decline was not permanent and the average number of likes gradually increased again over the next few days. We surmise that collusion networks gradually replenish their access token pool with fresh access tokens from new and returning users.

To mitigate this, we next invalidated all access tokens that were observed till day 28 (marked by red cross) and also

Figure 4. The impact of our countermeasures on two popular collusion networks. We observed that collusion network activities were not impacted by the reduction in access token rate limit. Although access token invalidation significantly reduced collusion network activities, it could not completely stop them. Clustering-based access token invalidation also did not help. Our IP rate limits effectively countered most collusion networks that used a few IP addresses. We targeted autonomous systems (ASes) of collusion networks that used a large pool of IP addresses.



(a) hublaa.me



(b) official-liker.net

began invalidating 50% of newly observed access tokens on a daily basis (marked by orange cross). We observed a sharp decline for both hublaa.me and official-liker.net on day 28 when we invalidated all access tokens. However, average likes by hublaa.me started to bounce back and those by official-liker.net stabilized at 100 over the next few days. We suspect that the rate of fresh access tokens from new and returning users exceeded our rate of daily access token invalidation. This is due to the rather small number of distinct new colluding accounts milked daily by our honeypots.

To increase our access token invalidation rate, starting day 36, we began invalidating all newly observed access tokens on a daily basis as marked by blue crosses in Figure 4. We observed a steady decrease in average likes by hublaa.me from day 36 to day 44. The hublaa.me's site was temporarily shut down on day 45. The site resumed operations on day 51 and their average number of likes decreased to 120. The official-liker.net sustained their likes between 110 and 192 despite our daily access token invalidation. Although regular access token invalidation curbed collusion network activities, we conclude that it cannot completely stop them because honeypot milking can only identify a subset of all newly joining users. Therefore, we decided not to pursue regular access token invalidation further.

5.3. Temporal clustering

Collusion networks provide likes on submitted posts in less than 1 minute. Such bursts of liking activity can be detected by temporal clustering algorithms[12, 16] which are designed to detect accounts that act similarly at around the same time for a sustained period of time. Starting day 55, as marked by cyan squares in Figure 4, we used SynchoTrap[12] to cluster synchronized access token abuse by collusion network accounts. Surprisingly, we did not observe any major impact on collusion network activities. Our drill-down analysis shows that collusion networks avoided detection by (1) using a different set of accounts to like target posts and (2) spreading out liking activities performed by each access token over time. Figure 5 shows that different sets of accounts liked posts of our honeypot accounts. We note that 76 and 30% accounts liked at most one post of our honeypots for hublaa.me and official-liker.net, respectively. Figure 6 shows that collusion networks did not binge use our honeypot accounts within a short timeframe. We note that the hourly average of likes performed by our honeypot accounts ranges between 5 and 10. Therefore, collusion network accounts did not behave similarly at around the same time for a sustained period of time.

Figure 5. Number of our honeypot posts liked by collusion network accounts. We observed that a small fraction of collusion network accounts like multiple honeypot posts.

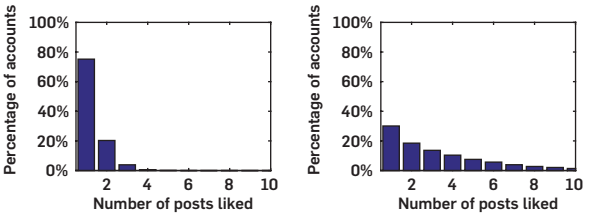


Figure 6. Hourly time series of number of likes performed by our honeypot accounts. We observed that collusion networks spread out liking activities performed by our honeypot accounts over time.

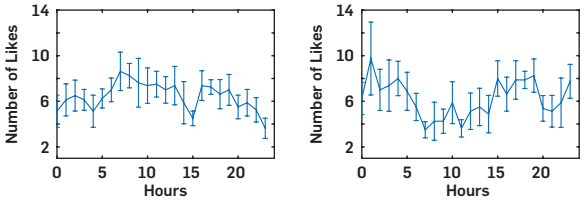
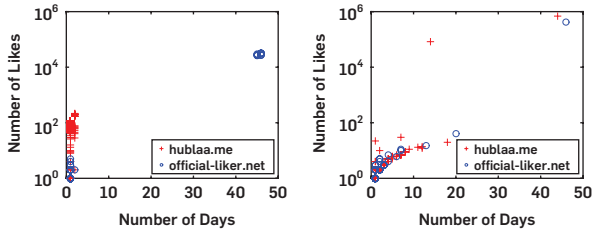


Figure 7. Source IP addresses and ASes of Facebook Graph API requests by hublaa.me and official-liker.net to like posts of our honeypot accounts.



5.4. IP- and AS-based limits

We next targeted the origin of collusion networks to further mitigate their activities. To this end, we tracked the source IP addresses of the Facebook Graph API requests for liking posts of our honeypot accounts. Figure 7(a) shows the scatter plot of these IP addresses where x-axis represents the number of days an IP address was observed during our countermeasures and y-axis represents the total number of likes generated by each IP address. It is noteworthy that a few IP addresses accounted for a vast majority of likes for official-liker.net. Therefore, we imposed a daily and weekly IP rate limits on the like requests beginning day 46. Note that this rate limit will not impact activities of normal users (e.g., regularly accessing Facebook via commodity web browsers) because this IP rate limit is only applicable to like requests by the Facebook Graph API using access tokens. Figure 4 shows that official-liker.net stopped working immediately after we imposed IP rate limits. Although not shown in Figure 4 due to space constraints, other popular collusion networks in Table 2 also stopped working on day 63. The only exception is hublaa.me, which used a large pool of more than 6000 IP addresses and circumvented the IP rate limits. Further analysis in Figure 7(b) reveals that all of hublaa.me's IP addresses belonged to two distinct autonomous systems (ASes) of bulletproof hosting providers.[9] On day 70, we started to block like requests from these ASes for susceptible applications which helped in ceasing all likes from hublaa.me. Note that we targeted a small set of susceptible applications for AS blocking to mitigate the risk of collateral damage to other applications.

5.5. Limitations

First, our countermeasures should not result in collateral damage while being robust to evasion attempts by collusion networks. To date, we have not received any collateral damage complaints from popular third-party developers. Therefore, we conclude that our countermeasures do not result in significant false positives. Second, our countermeasures need to be robust against potential evasion attempts by collusion networks. Our countermeasures have proven to be long-lasting for several months now. In future, collusion networks can try to evade our countermeasures in several ways. For example, collusion networks can use many different IP addresses and ASes (e.g., using botnets and proxies) to circumvent our IP- and AS-based countermeasures. If and when that happens, we can again use honeypots to swiftly identify IP addresses and ASes used by collusion networks. Third, collusion networks may try to identify the honeypot accounts that we use to infiltrate them. For example, collusion networks can try to detect our honeypot accounts which currently make very frequent like/comment requests. To circumvent such detection, we can create multiple honeypot accounts to decrease the frequency of per-account like/comment requests.

6. CONCLUSION

We presented a comprehensive measurement study of collusion-based reputation manipulation services on Facebook. Our results raise a number of questions that motivate future research. First, we would like to investigate potential access token leakage and abuse on other popular online services that implement OAuth 2.0. For instance, YouTube, Instagram, and SoundCloud implement OAuth 2.0 to support third-party applications. Second, in addition to reputation manipulation, attackers can launch other serious attacks using leaked access tokens. For example, attackers can steal personal information of collusion network members as well as exploit their social graph to propagate malware. We plan to investigate other possible attacks as well. Third, although our simple countermeasures have been effective now for more than 6 months, collusion networks may start using more sophisticated approaches to evade them in future. We plan to investigate more sophisticated machine learning-based approaches to robustly detect access token abuse. We are also interested in developing methods to detect and remove reputation manipulation activities of collusion network members. Finally, a deeper investigation into the economic aspects of collusion networks may reveal operational insights that can be leveraged to limit their financial incentives.

Acknowledgments

This work is supported in part by the National Science Foundation under grant number CNS-1715152 and by an unrestricted gift from Facebook.

References

1. App Review. https://developers.facebook. com/docs/apps/review.
2. Death By Captcha | Best and cheapest captcha service. http://www.deathbycaptcha.com/.
3. Facebook—Annual Report. http://investor.fb.com/secfiling.cfm?filingID=1326801-16-43&CIK=1326801.
4. Facebook Login for Apps—Developer Documentation. https://developers.facebook.com/docs/facebook-login.
5. Manually Build a Login Flow—Facebook Login. https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow.
6. OAuth 2.0 Threat Model and Security Considerations. https://tools.ietf.org/html/rfc6819.
7. Securing Graph API Requests. https://developers.facebook.com/docs/graph-api/securing-requests.
8. Twitter Inc.—Quarterly Report. http://files. shareholder.com/downloads/AMDA-2F526X/3022492942x0xS1564590-16-21918/1418091/filing.pdf, August 2016.
9. Alrwais, S., Liao, X., Mi, X., Wang, P., Wang, X., Qian, F., Beyah, R., McCoy, D. Under the shadow of sunshine: Understanding and detecting bulletproof hosting on legitimate service provider networks. In *IEEE Symposium on Security and Privacy* (2017).
10. Bilge, L., Strufe, T., Balzarotti, D., Kirda, E. All your contacts are belong to us: Automated identity theft attacks on social networks. In *WWW* (2009).
11. Boshmaf, Y., Logothetis, D., Siganos, G., Leria, J., Lorenzo, J., Ripeanu, M., Beznosov, K. Integro: Leveraging victim prediction for robust fake account detection in OSNs. In *Network and Distributed System Security Symposium* (2015).
12. Cao, Q., Yang, X., Yu, J., Palow, C. Uncovering large groups of active malicious accounts in online social networks. In *ACM Conference on Computer and Communications Security* (2014).
13. Cristofaro, E.D., Friedman, A., Jourjon, G., Kaafar, M.A., Shafiq, M.Z. Paying for likes? Understanding Facebook like fraud using honeypots. In *ACM Internet Measurement Conference (IMC)* (2014).
14. Hardt, E.D. The OAuth 2.0 authorization framework. In *IETF RFC 6749* (October 2012).
15. Fett, D., Kusters, R., Schmitz, G. A comprehensive formal security analysis of OAuth 2.0. In *ACM Conference on Computer and Communications Security* (2016).
16. Jian, M., Cui, P., Beutel, A., Faloutsos, C., Yang, S. CatchSync: Catching synchronized behavior in large directed graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014).
17. Shernan, E., Carter, H., Tian, D., Traynor, P., Butler, K. More guidelines than rules: CSRF vulnerabilities from noncompliant OAuth 2.0 implementations. In *Proceedings of the International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA)* (2015).
18. Stein, T., Chen, E., Mangla, K. Facebook immune system. In *Workshop on Social Network Systems (SNS)* (2011).
19. Stringhini, G., Wang, G., Egele, M., Kruegel, C., Vigna, G., Zheng, H., Zhao, B.Y. Follow the green: Growth and dynamics in Twitter follower markets. In *ACM Internet Measurement Conference (IMC)* (2013).
20. Sun, S.-T., Beznosov, K. The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems. In *ACM Conference on Computer and Communications Security* (2012).
21. Thomas, K., McCoy, D., Grier, C., Kolcz, A., Paxson, V. Trafficking fraudulent accounts: The role of the underground market in Twitter spam and abuse. In *USENIX Security Symposium* (2013).
22. Viswanath, B., Bashir, M.A., Crovella, M., Guha, S., Gummadi, K.P., Krishnamurthy, B., Mislove, A. Towards detecting anomalous user behavior in online social networks. In *USENIX Security Symposium* (2014).
23. Wang, G., Wang, T., Zheng, H., Zhao, B.Y. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *USENIX Security Symposium* (2014).
24. Webb, S., Caverlee, J., Pu, C. Social honeypots: Making friends with a spammer near you. In *Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)* (2008).
25. Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F. SybilLimit: A near-optimal social network defense against Sybil attacks. In *IEEE Symposium on Security and Privacy* (2008).

Shehroze Farooqi and Zubair Shafiq ({shehroze-farooqi, zubair-shafiq}@uiowa.edu), The University of Iowa, Iowa City, IA, USA.

Nektarios Leontiadis (leontiadis@fb.com), Facebook, Washington, D.C., USA.

Fareed Zaffar (fareed.zaffar@lums.edu.pk), Lahore University of Management Sciences, Lahore, Pakistan.