Sybil-Resilient Online Content Rating

Dinh Nguyen Tran, Bonan Min, Jinyang Li, Lakshminarayanan Subramanian
Abstract

• Vote Aggregation Problem
• Sybil Attack
• Adaptive vote flow aggregation
• Vote-history-based restriction
• Evaluation
Content Pollution

• Several Incidents of contamination in user-content hosting sites

• Rating or Ranking the content in OSN or P2P is very important!

• Famous algorithms: PageRank or HITS
  – Link Structure
  – Do not apply to videos, news articles, reviews or documents.
  – In OSN / P2P: votes are promising
Sybil Attack

- Votes from Sybil nodes can out-vote votes from real users.
  - The famous Slashdot poll.
  - Companies in Youtube vote for their videos.
Assumption

• **Sumup leverages trust links embedded in social networks.**
  – You need human efforts to become friends of others.
  – Attackers can hardly obtain large number of trust links.
  – Attackers can construct links among Sybil nodes as they like.
Related Work (1/3)

- **Hyperlink-based ranking: PageRank, HITS**
  - A links to B: A votes for B
  - A page with higher ranking can be linked by higher pages.
  - Sybil Attack: Sybil pages links to each other and then they link to A \(\Rightarrow\) improve A’s ranking
  - Solution for Sybil: introducing a small set of trusted web pages with a small probability.(15%).
Related Work (2/3)

- **History-based reputation**
  - Famous Model: EigenTrust, Credence
  - Limitations:
    - Cold-start problem
    - Tricky attack
    - Sybil Attack
Related Work (3/3)

• **Trust network based reputations**
  – Trust link: offline social relationship between users.
  – Famous: Advogato, Appleseed, Sybilproof

• **Sybil defense using trust networks**
  – SybilGuard
  – SybilLimit
  – Ostra
  – Kaleidoscope
Vote Aggregation Scenario

• Scenario.

• Sybil nodes can create symmetric sub-graphs like that created by honest peers.

• We need a source node.

• A central entity can maintain all the information, and use some algorithms to defend against Sybil nodes.
Vote Aggregation Problem

- **Vote Aggregation Problem**

  - **Assumptions:**
    - $G = (V, E)$ be a trust network, $s \in V$
    - $V_h$ is a subset of honest users and $s \in V_h$
    - $e_A$ is the number of attack edges from $V - V_h$
    - nodes in $V$ cast votes on object $O$

  - **A vote aggregation Mechanism achieves three properties:**
    - Aggregate all votes from honest users.
    - Limit the number of bogus votes from Sybil nodes by $e_A$.
    - Ignore votes from nodes that repetitively cast bogus votes.
Basic Max flow (1/2)

- **attack capacity**: maximum number of votes cast by Sybil nodes.

**Figure**: SumUp computes a set of approximate max-flow paths from the vote collector $S$ to all voters ($A, B, C, D$). Straight lines denote trust links and curly dotted lines represent the vote flow paths along multiple links. Vote flow paths to honest voters are "congested" at links close to the collector while paths to Sybil voters are also congested at far-away attack edges.
Basic Max flow (2/2)

• **Limitations:**
  – Trust network is sparse.
  – Total trust votes can be collected is small.
  – YouTube and Flickr have small median node degrees.

• **Solutions:**
  – Increase capacity of each link
    → increase capacity of attackers.
  – Tradeoff: honest votes VS bogus votes
Adaptive Vote Flow Technique

• **Observations:**
  – The number of honest votes is smaller than total users.
  – Honest users are more spread than Sybil nodes behind Sybil nodes.

• **Key aims:**
  – Restrict the maximum number of votes to $C_{\text{max}}$.
  – Capacity assignment to each edge $\rightarrow$ solving the tradeoff.
  – Leverage user history (feedback mechanism)
Capacity assignment

- Source distributes $C_{\text{max}}$ tickets
- Attack edges beyond the envelope are assigned 1
- BFS-like distribution for tickets.
Capacity Assignment Algorithm

• **Pruning**
  – reduce the number of attack edges

• **Assign positive capacity values**
  – To collect most of $C_{\text{max}}$ votes.
  – Minimize attack capacity.

• **max-flow computation**
  – a fast approximation algorithm that incrementally updates the vote count
Pruning(1)

- **Property**
  - Bound the in-degree of each node to constant value: $d_{\text{in-thres}}$.
  - Keep honest nodes cast one vote.
  - Keep connecting.

- **Tree topology:**
  - Limit Sybil nodes.
  - Also limit honest nodes.
  - Total votes: $(1 - \frac{1}{e})C_{\text{max}}$
Pruning(2)

**Steps**

1. Remove all links except those connecting level l and level l+1.
2. Remove incoming links at each node so that the remaining links do not exceed $d_{\text{in\_thres}}$.
3. Add back links removed in step 1 for nodes with less than $d_{\text{in\_thres}}$ incoming edges.
4. Add one outgoing link back to nodes without outgoing links.
Capacity assignment (1/2)

• **Goals:**
  – Collect almost all votes if less than $C_{\text{max}}$ honest votes.
  – Minimize the attack capacity, $C_A \leq e_A$.
  – Construct a vote envelope.
    • Attack edge beyond envelope: only 1 capacity
    • Minimize the assignment for edges in envelope.
Capacity assignment (2/2)
Incremental vote collection

• **Exist max-flow algorithms do not work.**
  – Do not care about the specific paths.
  – Can not update subsequent votes.

• **Ford-Fulkerson Algorithm**
  – O(E) running time.
  – BFS

• **SumUp’s algorithm**
  – DFS from votes to source
  – Number of time (300 steps)
  – Easier to find a path.
Security Properties

- Expected capacity per attack edge is:
  \[ \frac{E(C_A)}{e_A} = 1 + O\left(\frac{C_{\text{max}} \log C_{\text{max}}}{n}\right) \]

- Suppose \( C_{\text{max}} \) random honest voters, the expected faction of votes collected is:
  \[ 1 - o(1) \text{ if } C_{\text{max}} = O(n^{1/2}) \]
Practical Issues

• **Find n**
  – Expanded ring search approach.
  – Vote collector can fix a diameter and consider nodes with the diameter.
  – Impose an degree bound for each node.

• **Find** $C_{\text{max}}$
  – $n^{1/2}$ guarantees the security properties.
Exceptions

• Sybil nodes near source
• CA is too big.
• History can be used
Leveraging user feedback

• **Reduce the assignment values for the path, like Ostra.**

• **Do for each node as a source**
  - Adjust each link’s capacity assignment.
    • Distributes the tickets evenly according to the weights.
    • Same increase in negative histories $\rightarrow$ distribution keeps the same.

\[
\frac{w(h_{i,j})}{w(h_{i,k})} = \frac{w(h_{i,j} + h')}{w(h_{i,k} + h')}
\]

\[w(h_{i,j}) = a \times b^{h_{i,j}}\]

  - Delete the links whose negative history exceeding a certain threshold.
    • Delete the link if the number of negative history is five times as the assigned capacity.
    • Add back previous pruned incoming edges.
Advantages of deleting edges

• **Sybil nodes**
  – finally be restricted

• **Honest nodes**
  – use another path to reach source nodes
  – Will not be impacted by Sybil nodes in the downstream links
Evaluation

• **Results**
  – Collected no more than eA bogus votes with collecting 80% votes from Cmax voters.
  – Feedback mechanism works.
  – Many false popular articles in Digg.
1. For all networks under evaluation, SumUp bounds the average number of bogus votes collected to be no more than $e^A$ while being able to collect > 90% of honest votes when less than 1% of honest users vote.

2. By incorporating feedback from the vote collector, SumUp dramatically cuts down the attack capacity for adversaries that continuously cast bogus votes.

3. We apply SumUp to the voting trace and social network of Digg, a news aggregation site that uses votes to rank user-submitted news articles. SumUp has detected hundreds of suspicious articles that have been marked as "popular" by Digg. Based on manual sampling, we believe at least 50% of suspicious articles found by SumUp exhibit strong evidence of Sybil attacks.

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Edges</th>
<th>Degree</th>
<th>Directed?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\times 1000$</td>
<td>$\times 1000$</td>
<td>50% (90%)</td>
<td></td>
</tr>
<tr>
<td>YouTube [18]</td>
<td>446</td>
<td>3,458</td>
<td>2 (12)</td>
<td>No</td>
</tr>
<tr>
<td>Flickr [17]</td>
<td>1,530</td>
<td>21,399</td>
<td>1 (15)</td>
<td>Yes</td>
</tr>
<tr>
<td>Synthetic [24]</td>
<td>3000</td>
<td>24,248</td>
<td>6 (15)</td>
<td>No</td>
</tr>
</tbody>
</table>
Figure 4: The average capacity per attack edge as a function of the fraction of honest nodes that vote. The average capacity per attack edge remains close to 1, even if 1/10 of honest nodes vote.
Figure 5: The fraction of votes collected as a function of fraction of honest nodes that vote. SumUp collects more than 80% votes, even 1/10 honest nodes vote.
Figure 6: The fraction of votes collected for different $d_{in\_thres}$ (YouTube graph). More than 90% votes are collected when $d_{in\_thres} = 3$. 
Figure 7: Average attack capacity per attack edge decreases as the number of attack edges per adversary increases.
Figure 8: The fraction of votes collected for different threshold for non-greedy steps. More than 70% votes are collected even with a small threshold (10) for non-greedy steps.
Figure 9: The running time of one vote collector gathering up to 1000 votes. The Ford-Fulkerson max-flow algorithm takes 50 seconds to collect 1000 votes for the YouTube graph.
Figure 10: Average attack capacity per attack edge as a function of voters. SumUp is better than SybillLimit in the average case.

Figure 11: The change in attack capacity as adversaries continuously cast bogus votes (YouTube graph). Capacity adjustment and link elimination dramatically reduce $C_A$ while still allowing SumUp to collect more than 80% of the honest votes.
Defending Digg against Sybil attacks

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Nodes</td>
<td>3,002,907</td>
</tr>
<tr>
<td>Number of Edges</td>
<td>5,063,244</td>
</tr>
<tr>
<td>Number of Nodes in SCC</td>
<td>466,326</td>
</tr>
<tr>
<td>Number of Edges in SCC</td>
<td>4,908,958</td>
</tr>
<tr>
<td>Out degree avg (50%, 90%)</td>
<td>10 (1, 9)</td>
</tr>
<tr>
<td>In degree avg (50%, 90%)</td>
<td>10 (2, 11)</td>
</tr>
<tr>
<td>Number of submitted (popular) articles 2004/12/01-2008/09/21</td>
<td>6,494,987 (137,480)</td>
</tr>
<tr>
<td>Diggs on all articles avg (50%, 90%)</td>
<td>24 (2, 15)</td>
</tr>
<tr>
<td>Diggs on popular articles avg (50%, 90%)</td>
<td>862 (650, 1810)</td>
</tr>
<tr>
<td>Hours since submission before a popular article is marked as popular avg (50%, 90%)</td>
<td>16 (13, 23)</td>
</tr>
<tr>
<td>Number of submitted (popular) articles with bury data available 2008/08/13-2008/09/15</td>
<td>38,033 (5,794)</td>
</tr>
</tbody>
</table>

**Figure 13:** The distribution of the fraction of diggs collected by SumUp over all diggs before an article is marked as popular.

**Table 2:** Basic statistics of the crawled Digg dataset. The strongly connected component (SCC) of Digg consists of 466,326 nodes.
Figure 12: Distribution of diggs for all popular articles before being marked as popular and for all articles within 24 hours after submission.
<table>
<thead>
<tr>
<th>Factor</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold of the fraction of collected diggs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># of suspicious articles</td>
<td>41</td>
<td>131</td>
<td>300</td>
<td>800</td>
</tr>
<tr>
<td>Advertisement</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Phishing</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Obscure political articles</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Many newly registered voters</td>
<td>11</td>
<td>7</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Fewer than 50 total diggs</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>No obvious attack</td>
<td>10</td>
<td>14</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 3: Manual classification of 30 randomly sampled suspicious articles. We use different thresholds of the fraction of collected diggs for marking suspicious articles. An article is labeled as having many new voters if >30% of its votes are from users who registered on the same day as the article’s submission date.

Figure 14: The average number of buries an article received after it was marked as popular as a function of the fraction of diggs collected by SumUp before it is marked as popular. The Figure covers 5,794 popular articles with bury data available.
CONCLUSION

• This paper presented SumUp, a content voting system that leverages the trust network among users to defend against Sybil attacks.

• We demonstrate the real-world benefits of SumUp by evaluating it on the voting trace of Digg: SumUp detected many suspicious articles marked as “popular” by Digg.

• We have found strong evidence of Sybil attacks on many of these suspicious articles.