SybilBF: Defending against Sybil Attacks via Bloom Filters

Hengkui Wu, Dong Yang, and Hongke Zhang

Distributed systems particularly suffer from Sybil attacks, where a malicious user creates numerous bogus nodes to influence the functions of the system. In this letter, we propose a Bloom filter-based scheme, SybilBF, to fight against Sybil attacks. A Bloom filter presents a set of Sybil nodes according to historical behavior, which can be disseminated to at least $n\cdot(e-1)/e$ honest nodes. Our evaluation shows that SybilBF outperforms state of the art mechanisms improving SybilLimit by a factor of $(1/e)\gamma$ at least.

Keywords: Sybil attack, Bloom filter, security.

I. Introduction

Distributed systems particularly suffer from Sybil attacks, in which a malicious user creates numerous fake identities (called Sybil nodes) to control a large fraction of the system [1]. Examples of such systems include file sharing, email, instant messaging, DHT routing, social network service, online voting, reputation systems, and so on. There are two categories of schemes to defend against Sybil attacks: centralized and distributed. The centralized schemes use a trusted central authority acting as the admission control system to limit Sybil attacks. However, lack of widely accepted central authority and the single point of failure make the centralized approaches impractical. Moreover, many users would not like to supply sensitive personal information for Internet services.

The Sybil attack problem should be resolved in distributed ways. Recently, some researchers leverage social networks to mitigate Sybil attacks [2]-[5], which are considered as state of the art promising defense schemes.

Social network-based Sybil defense schemes are based on the assumption that a malicious user can create arbitrary numerous Sybil nodes but cannot establish arbitrary trust edges with honest nodes. They try to find the minimum cut between the Sybil region and the non-Sybil region. The main idea of social network-based defense schemes is to bind the number of introduced Sybil nodes through the limited attack edges (a trust edge between a Sybil node and an honest node is called an attack edge). For example, the number of Sybil nodes per attack edge introduced in SyiblGuard [2] is $O(\sqrt{n}\log n)$. In SybilLimit [3], it is $O(\log n)$, where n is the number of honest nodes.

In this letter, we propose a Bloom filter-based Sybil-resilient scheme to fight against Sybil attacks. An honest node uses a Bloom filter to summarize a set of Sybil nodes based on historical behavior, which is called a pattern. A few patterns from different nodes can be aggregated into one pattern using an improved OR-based aggregation algorithm. Finally, every pattern is disseminated in the system, where each pattern could be propagated to at least $n\cdot(e-1)/e$ honest nodes in SybilBF. We evaluate the performance of the system and the results show that SybilBF outperforms state of the art algorithms, which improves SybilLimit by a factor of $(1/e)^{\gamma}$ at least.

II. SybilBF Model

1. Using a Bloom Filter to Summarize a Set of Sybil Nodes

A node can get a list of Sybil nodes via historical behavior, based on which we use Bloom filters [6] as indices in SybilBF. A Bloom filter is a space-efficient data structure for presenting

Manuscript received Nov. 11, 2010; revised Dec. 26, 2011; accepted Jan. 10, 2011.

Hengkui Wu (phone: +8615192510211, email: hkwu.bjtu@gmail.com) is with the Science and Technology on Electronic Test & Measurement Laboratory, The 41st Research Institute of CETC, Qingdao, China, and also with the School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China.

Dong Yang (email: dyang@bjtu.edu.cn) and Hongke Zhang (email: hkzhang@bjtu.edu.cn) are with the School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China.

http://dx.doi.org/10.4218/etrij.11.0210.0431

a set and answering membership queries. This space efficiency is achieved at the expense of a small possibility of false positive in membership queries. In most conditions, the space saving outweighs the drawback when the possibility of false positive is tuned sufficiently low.

A Bloom filter is a set $S=\{s_1, s_2,..., s_q\}$ of q Sybil nodes. In the beginning, it is an m-bit all-zero array. When inserting a Sybil node s into the set S, a Bloom filter uses k independent hash functions, $h_1, h_2,..., h_k$, to map each Sybil node to values in the range $\{1,..., m\}$. The corresponding bits of $h_i(s)$ will be set to 1. To check whether an element s belongs to the set S expressed by the Bloom filter, one just needs to check whether all $h_i(s)$ are set to 1. If so, s is a member of S (with a small possibility it is wrong, which is called false positive), otherwise it is not.

We assume the hash functions are totally random and $kq<m$. When all the Sybil nodes in the set S are hashed to the m-bit array by the k random hash functions, the possibility that one of m positions in the array still equals to 0 is

$$p = (1-1/m)^{kq} \approx e^{-kq/m}. \quad (1)$$

Therefore, the false positive possibility f can be expressed as

$$f = (1-p)^k = (1-e^{-kq/m})^k. \quad (2)$$

In an efficient Bloom filter-based system, the false positive possibility should be minimized. From (2), we can get that when $k=\ln 2\cdot(m/q)$, f is minimized and $f_{\min}\approx(0.6185)^{m/q}$.

2. Pattern Aggregation

A Bloom filter summarizing a set of Sybil nodes is called a pattern. Different patterns from numerous nodes can be aggregated into one pattern according to (3). We improve OR-based aggregation by adding a new bit X, and when $a_i\neq b_i$, this bit will be set to X:

$$j_i = \begin{cases} a_i & \text{if } a_i = b_i \\ X & \text{otherwise,} \end{cases} \quad (3)$$

$$L(A,B) = \frac{C - \varepsilon\times D + \phi\times E + \psi\times F}{\left|A^0\right| + \left|A^1\right| + \left|A^X\right|}. \quad (4)$$

The similarity of A and B in Table 1 can be noted in (4). Here, $|A^m|=\{a_i^m | a_i^m\in\{0,\ 1\}\wedge m\in\{0,\ 1,\ X\}\wedge j_i=m\Rightarrow j_i^m=1\}$, which indicates the number of m (0, 1, or X) in A. In order to make the aggregation efficient, we minimize the number of X-bits through tuning $L(A, B)$. The pattern aggregation mechanism is a plug-in module in SybilBF. Different approaches can be used in SybilBF, but all the users in the system should use the same scheme. The aggregation approaches could also be based on counting Bloom filters, dynamic Bloom filters [7], and so on.

Table 1. Similarity of two patterns.

a_i	b_i	Coeff.	Term	Computed as
0 1	0 1	1	C	$\left\|A^0\wedge B^0\right\|+\left\|A^1\wedge B^1\right\|$
0 1	1 0	ε	D	$\left\|A^0\wedge B^1\right\|+\left\|A^1\wedge B^0\right\|$
0 1 X X	X X 0 1	ϕ	E	$\left\|A^x\oplus B^x\right\|$
X	X	ψ	F	$\left\|A^x\wedge B^x\right\|$

3. Pattern Propagation

A propagating user Z disseminates pattern copies through the system. According to the distance (with the shortest path) to the propagating user, every node is considered in different levels. For example, node Z' is i hops (with the shortest path) from the propagating user, so node Z' is in level i. Z sends z pattern copies to its neighbors (in level 1), such as $z/2$ for Y and $z/2$ for Y', where Y and Y' are node Z''s neighbor nodes in level 1. Y remains one of the copies and sends the other $z/2-1$ copies to its neighbors in the next level. If the node does not have neighbors in the next level, it will destroy and drop the pattern copies.

We assume social network G is an expander random graph [8]. Let β denote G's expander factor. Let η denote G's diameter. Let d denote the average node degree. Let N_i denote the number of nodes at level i for a propagating user. For the expander graph G, there exists a value η' to make the following equations stand:

$$\frac{N_i}{N_{i-1}} \geq \beta \quad \forall i\in[0,\eta'), \quad (5)$$

$$\frac{N_{i-1}}{N_i} \geq \beta \quad \forall i\in[\eta',\eta). \quad (6)$$

We assume that a propagating user propagates μn pattern copies for $\mu = e^{1/(1-\beta)}\cdot e^{\beta/(\beta+1)} = e^{(\beta^2+1)/(\beta^2-1)}$. Let M_i denote the number of pattern copies sent from level i to level $i+1$. Let p_i denote the probability a node at level i connects with some nodes at the next level $i+1$. Therefore, if the pattern copies are not used out in level i, we can use the following equation to describe M_i:

$$M_i = \begin{cases} \mu n & \text{if } i = 0 \\ p_i (M_{i-1} - N_i) & \text{if } i \geq 1. \end{cases} \quad (7)$$

The probability that a node at level i does not connect to any node at level i+1, namely 1–p_i, can be described as

$$1 - p_i = \left(\frac{N_{i-1} + N_i - 1}{N_{i-1} + N_i + N_{i+1} + \ldots + N_{\eta-1}}\right)^d. \quad (8)$$

Because d>1, when $i < \eta'$, we get

$$\begin{aligned} \forall i \in [1, \eta') \quad 1 - p_i &< \frac{N_{i-1} + N_i - 1}{N_{i-1} + N_i + N_{i+1} + \ldots + N_{\eta-1}} \\ &< \frac{1}{1 + \beta^{\eta'-i}} \Rightarrow \\ p_i > 1 - \frac{1}{1+\beta^{\eta'-i}} \Rightarrow \ln(p_i) &> \frac{-1}{\beta^{\eta'-i}} \Rightarrow \\ \sum_{i=1}^{\eta'-1} \ln(p_i) &> \sum_{i=1}^{\eta'-1} \frac{-1}{\beta^{\eta'-i}} \Rightarrow \\ \sum_{i=1}^{\eta'-1} \ln(p_i) > \frac{-1}{\beta-1} \Rightarrow \prod_{i=1}^{\eta'-1} p_i &> e^{\frac{-1}{\beta-1}}. \end{aligned} \quad (9)$$

When $\eta' \leq i < \eta$, we can also get the following equation:

$$\begin{aligned} \forall i \in [\eta', \eta) \quad 1 - p_i &< \frac{N_{i-1} + N_i - 1}{N_{i-1} + N_i + N_{i+1} + \ldots + N_{\eta-1}} \\ &< \frac{\beta^{\eta-i}}{1 + \beta^{\eta-i}} \Rightarrow \\ p_i > 1 - \frac{\beta^{\eta-i}}{1+\beta^{\eta-i}} \Rightarrow \ln(p_i) &> \frac{-\beta^{\eta-i}}{1+\beta^{\eta-i}} \Rightarrow \\ \sum_{i=\eta'}^{\eta-1} \ln(p_i) &> \sum_{i=\eta'}^{\eta-1} \frac{-\beta^{\eta-i}}{1+\beta^{\eta-i}} \Rightarrow \\ \sum_{i=\eta'}^{\eta-1} \ln(p_i) > \frac{-\beta}{\beta+1} \Rightarrow \prod_{i=\eta'}^{\eta-1} p_i &> e^{\frac{-\beta}{\beta+1}}. \end{aligned} \quad (10)$$

According to (9), (10), and $\mu = e^{(\beta^2+1)/(\beta^2-1)}$, we can easily derive

$$\begin{aligned} &\prod_{i=1}^{\eta-1} \frac{1}{p_i} < \mu \Rightarrow \prod_{i=1}^{\eta-1} \frac{n}{p_i} < \mu n \Rightarrow \\ &\frac{\sum_{i=1}^{\eta} p_i}{\prod_{i=1}^{\eta-1} p_i} < M_0 \Rightarrow \sum_{i=1}^{\eta} \frac{p_i}{p_1 p_2 \cdots p_{\eta-1}} < M_0. \end{aligned} \quad (11)$$

For a random propagating user, we will calculate the possibility that a random node w at level i does not get the pattern copy. The possibility that a pattern copy is sent from level i to level i+1 is M_{i-1}/M_0, and the possibility it does not reach the level i is 1–M_{i-1}/M_0. So, the possibility that w does not get the pattern copy is 1–$M_{i-1}/(M_0N_i)$. Applying to all the M_0 pattern copies, the possibility will be $(1 - M_{i-1}/(M_0 N_i))^{M_0}$. The total number of nodes which do not get the pattern copies is

$$\sum_{i=1}^{\eta} (1 - \frac{M_{i-1}}{M_0 N_i})^{M_0} \cdot N_i < \frac{\sum_{i=1}^{\eta} L_i}{e^{M_{\eta'-1}/L\eta'}} < \frac{\sum_{i=1}^{\eta} L_i}{e} = \frac{n}{e}. \quad (12)$$

Therefore, the number of nodes which can get the pattern copies is at least (e–1)/e. A node, which gets a copy of the pattern, can recognize the Sybil nodes in the pattern, so the probability that a Sybil node in the pattern can be detected is at least (e–1)/e.

III. Evaluation

1. Introduced Sybil Nodes

A pattern can reach at least n·(e–1)/e nodes, so the probability that one node gets the information is at least (e–1)/e and the probability that the node does not get the information is 1–(e–1)/e=1/e. Knowing the Sybil node historical behavior, we assume an average number of γ nodes. For any Sybil node, the probability that its historical behavior is disseminated is 1–(1/e)$^{\gamma}$.

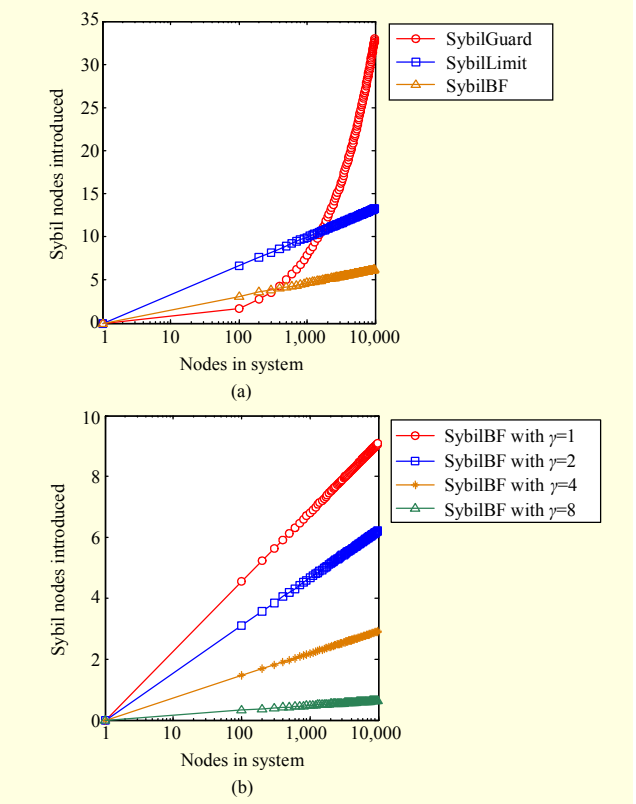

Fig. 1. Sybil nodes introduced in SybilGuard, SybilLimit, and SybilBF.

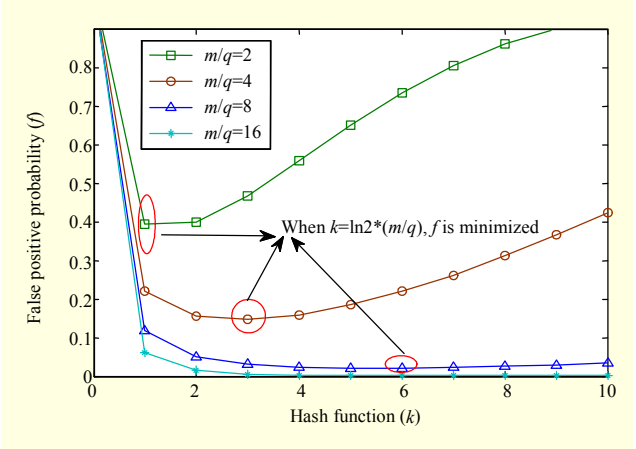

Fig. 2. False positive probability in SybilBF.

Figure 1(a) illustrates the Sybil nodes per attack edge introduced in SybilGuard, SybilLimit, and SybilBF. When the number of honest nodes in the system increases, the number of Sybil nodes introduced in SybilGuard increases faster than SybilLimit and SybilBF. SybilBF can improve SybilLimit by a factor of $(1/e)^{\gamma}$. When γ=4, if Sybil nodes introduced in SybilLimit is 1,000, it will be 18 in SybilBF. Figure 1(b) plots the Sybil nodes introduced in SybilBF with different γ. When γ is not sufficiently low, the number of Sybil nodes can be limited efficiently.

2. False Positive

SybilBF limits Sybil attacks based on the historical behavior of Sybil nodes. We use Bloom filters to present the set of Sybil nodes. The Bloom filter is a space-efficient data structure, however, there is a small probability of errors. We analyze the false positive probability in SybilBF.

Figure 2 plots the false positive probability with different hash functions and m/q values. When using the same number of hash functions, the false positive probability is in inverse proportion to m/q. When k=ln2(m/q), the false positive probability is minimized. These points are labeled by red rings in the figure. We can tune k and m/q to make the false positive probability sufficiently low.

IV. Conclusion

We proposed a Bloom filter-based scheme to fight against Sybil attacks. A node summarizes a set of Sybil nodes via a Bloom filter (called a pattern) based on historical behavior and disseminates it in the system. We proved that at least $n\cdot(e-1)/e$ honest nodes will get the pattern copy. We evaluated the system, and the results show that it improves the current optimal social network-based Sybil-resilient scheme by a factor of $(1/e)^{\gamma}$ at least. The results also show that we can tune parameters to make the false positive of the Bloom filter sufficiently low. The update of the dynamic set of Sybil nodes using the Bloom filter is the next step in our work.

References

[1] J.R. Douceur, "The Sybil Attack," *Proc. 1st Int. Workshop Peer-to-Peer Syst.*, Cambridge, MA, USA, Mar. 2002, pp. 251-260.

[2] H. Yu et al., "SybilGuard: Defending against Sybil Attacks via Social Networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, 2008, pp. 576-589.

[3] H. Yu et al., "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, 2010, pp. 885-898.

[4] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil Nodes Using Social Networks," *Proc. 16th Netw. Distrib. Syst. Security Symp.*, San Diego, California, USA, Feb. 2009.

[5] N. Tran et al., "Sybil-Resilient Online Content Voting," *Proc. 6th USENIX Symp. Netw. Syst. Design Implementation*, Boston, MA, USA, Apr. 2009, pp. 15-28.

[6] B. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, 1970, pp. 422-426.

[7] D. Guo et al., "The Dynamic Bloom Filters," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, 2010, pp. 120-133.

[8] J. Leskovec et al., "Statistical Properties of Community Structure in Large Social and Information Networks," *Proc. WWW*, Beijing, China, Apr. 2008, pp. 695-704.