Secure Decentralized Content Distribution Network

December 29, 2020

1 Introduction

Digital content distribution network provides convenient means for publishers to reach broader audiences. The technology gives people more exposure to information, but the platform also needs to preventing information from overwhelming the audience. Commercial companies have been known to collect content from publishers and to use machine learning for maximizing the entertainment to clients. For example, Youtube and Tiktok collect user meta-information information for selectively feeding video. As the result, some companies have gained huge market value for using two technologies successfully, be it the sport, news, or politics.

We note that the company motivated by the business goal is not the best entity to support the digital commons which are more relevant to public interests, and the best way to address this problem is to have a public digital content distributing platform designed from a P2P approach. For such decentralized platforms to be useful and eventually adapted by people, the platform should be secure against malicious actions and efficient to broadcast messages. After surveying the decentralized network layer protocols, we note that GossipSub[2] offers both fast message forwarding and security. We are interested in the security aspect, since a decentralized network is more vulnerable to adversarial actions. We look into two attacks trying to understand the security guarantee of the protocol. An eclipse Attack aims to silence a node completely or distort the view of the node by dominating all its connections, and such can be further extended to target a group of nodes and the entire network. In a covert Flash Attack, sybils nodes stay in the network and behave properly for a while until they all stop propagating messages to break the system.

We created a codebase to test the performance of such protocol under those two attacks. In the Flash attack under a cold boot condition (when sybils node hides in the network at the start of the network formation), the network can be partitioned into smaller pieces in the first several rounds, but after a heartbeat management cycle, the network is able to bring up a good shape which eliminates the sybil nodes completely. We then analyze the Eclipse attack by first implementing the minimal structure and initiating a simple attack. Then we

add the safety mitigation recommended from a third-party security company, and devise new attacks in the updated protocol. We found out that the mitigation is successful to defend against attack Graft Snipping and Love bombing attacks discussed in Issues B and C in [1]. Without the mitigation, we can easily eclipse the targets without powerful adversaries as long as there is a sufficiently large number of sybil nodes. Then for launching a successful attack, we make further reasonable assumptions about what power to give the adversary, they include a few number of computationally strong sybils connected in a fast internal network, observing the network history around the targets nodes, freezing some links among honest peers. The attack is successful when there is a small number of publishers, and sybils node can dominate most of the target's mesh connection when the number of publishers becomes large. We created a codebase of 4500 lines of codes to emulate the actual network with the adversary, the code is available at `https://github.com/bx3/secure-decentralized-cdn.git`.

2 GossipSub implementation

GossipSub consists of two major component: **Mesh construction** and **Score function**. There are three types of honest nodes in the network: publishers are nodes who broadcast the honest transaction to other nodes, lurkers are nodes who participate in the maintenance of the mesh but do not publish transactions, and gossip only nodes who only pull information and do not maintain the mesh. Both publishers and lurkers actively push new messages to their peers. A good mesh facilitates fast message delivery, high throughput bandwidth, and resilience to attacks.

2.1 Mesh maintenance

All mesh nodes maintain two states for mesh construction: the number of outgoing mesh connections, and the total number of connections. Mesh is maintained periodically using a heartbeat. Between two heartbeat, a mesh node can send Graft or Prune messages to other nodes for notification so that the receiver knows it has been added or pruned by the sender from the sender's view of mesh. All nodes can directly send messages to all other nodes regardless if they are connected on the mesh, and therefore a node may have to handle any number of Graft and Prune messages within the heartbeat interval. For keeping the mesh network in a reasonable shape, maintenance is performed at every heartbeat to adjust the number of connections, see Fig 1. In the maintenance algorithm, if a node first checks if it has too few nodes, the node sends Graft messages to peers that have non-negative scores. If a peer has too many mesh connection, the nodes remove low score peers and send Prune messages to reflect the change. Additional mitigation schemes are implemented on top of the framework for improving the resilience against attacks. For example, a node must maintain at least D_out outgoing mesh connections, so that it reduces the chance of being eclipsed by the adversaries. Each node has a background thread

responsible for generating heartbeat message periodically for notifying its peer about the maintenance. In our implementation, we use a synchronous heartbeat that all nodes execute at the same time.

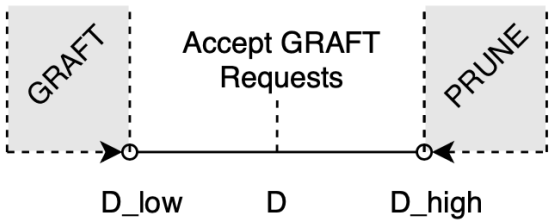

Figure 1: Mesh degree maintenance. D_low is usually set to 6, and D_high is 12 and D is 8

2.2 GossipSub score functions

Every node maintains its local view to all peers in the network by accumulating peer states based on past experience. A score function takes those states and output a scalar value judging the trustworthiness of peers. In the current implementation, we consider the following state properties:

- Time in the local mesh, how long a peer has stayed in the mesh previously.
- Responsiveness to deliver message, how many times the peer is the first to deliver messages.
- Mesh Delivery rate, how frequently a peer delivers messages. If a node does not meet the expectation, a negative penalty is added to the peer's score

3 Experiment Framework

To simulate and analyze the content distribution network (CDN), we build a comprehensive experiment framework including the network protocol itself which has been introduced in previous section, the network generator, and the network simulator with analysis and visualization.

3.1 Network generator

The network generator in the experiment framework can generate networks with various configurations and store in json file. The configurations include the number of publisher/lurk/sybil nodes, the bandwidth of up link and down link, the content generation interval (for publisher only), the initial peers number for each node, and if it is a cold boot system or not.

Figure 2 shows an example of a node in the network json file. The *id* is unique for each node. *Role* means the type of the node, and *0* indicates publisher. *Known* shows a list of all known nodes that could possibly build connection. *downB* and *upB* are download and upload bandwidth. And this node, as a publisher, has high probability to create new content every *interval* second. The experiment framework can read the network json file and run simulation.


```
{"id": 0,
 "role": 0,
 "known": [4, 7, 9, 12, 21, 25, 33, 34, 41, 49],
 "downB": 1000000.0,
  "upB": 1000000.0,
  "interval": 0.7
}
```


Figure 2: Example of a node in the network json file

3.2 Network simulator with analysis and visualization

We further build a network simulator to simulate the CDN under different types of attacks. Figure 3 shows the command line simulator interface. To initialize the simulation, one should offer the network json file at first. We can interact with the simulator in the command line interface to set simulation rounds and attack types. When the simulation ends, the simulator will show the visualization of the whole network, including all types of nodes and connections. The publisher, lurk and sybil nodes are in green, yellow, and red, respectively. The visualizer also has a zoom-in feature, where one can have a closer look at certain nodes and get more detailed information suck as peer scores on each neighbor. The simulator could also show some network statics changes during the simulation. In Figure 3, the analyzer shows the number of total transactions generated and received by the network. They can help us to analyze the latency and throughput.

4 Covert Flash Attack

We simulate two attacks on the network model, covert flash attack and eclipse attack. In the covert flash attack, sybil nodes connect to the network and behave properly for some period of time to build up the score. Then, they execute a coordinated network-wide attack whereby they stop propagating messages in an attempt to completely disrupt the network. In this attack, we assume adversarial nodes has no knowledge on the networks, they can only build connections to other nodes randomly.

First, we simulate this flash attack in a cold boot system with 5 publisher nodes, 25 lurk nodes, and 20 sybil nodes. Figure 4 visualizes the whole network

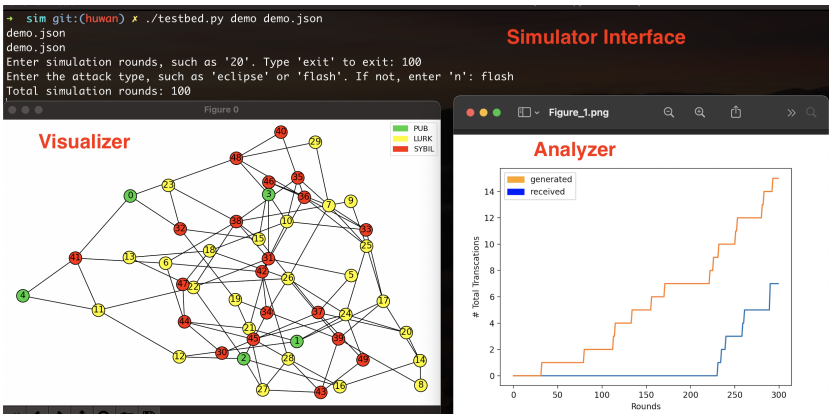

Figure 3: The network simulator. One can interact with the simulator in the command line interface to set simulation rounds and attack types. When simulation ends, the simulator will show the visualization of the whole network, including all type of nodes and connections. It will also show some statics changes during the simulation.

before and after the attack. Before the attack, sybil nodes are connected to some honest nodes randomly by behaving well for some period of time and building up the score, as shown on the left side of the figure. During the attack, all sybil nodes will not process or send any messages. As a result, the peer scores of adversaries will decrease since they can not earn any first send messages reward and will be punished because of the low message delivery rate. The right side of Figure 4 visualizes the network 200 rounds after the flash attack starts. All sybil nodes (red) are detected and pruned from the honest network.

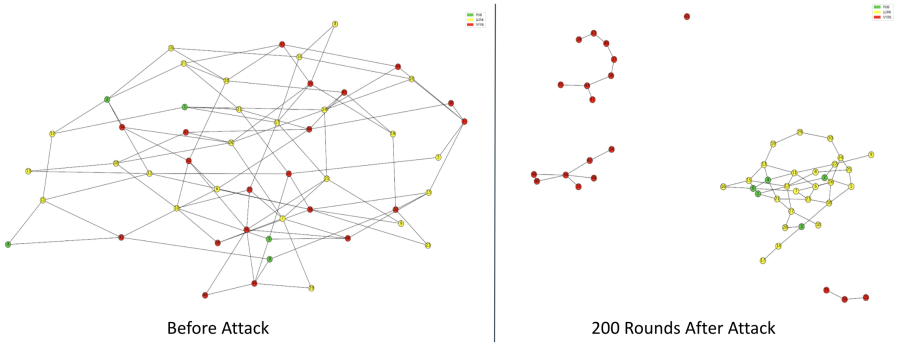

Figure 4: Network before and after the flash attack. All sybil nodes (red) are detected and pruned by the honest network.

4.1 The impact of different components in score function on flash attack

We examine the impact of different components in score function on flash attack. Figure 5 shows the received transactions of the network in flash attack.

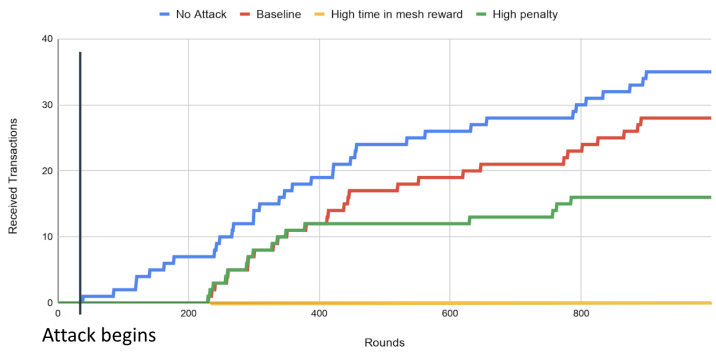

Figure 5: The received transactions of the network in flash attack. Changing the weight of different components in score function can reduce the ability to defense a flash attack.

The case without any attacks is shown in the blue line. The other 3 lines are the network with different weights in the score function. In the first 200 rounds during the attack, all of them will not receive any transactions. The whole network is disrupted by the attack. The red line is the baseline network with a fine-tuned score function. The weight of time in local mesh W_{P_1} is 0.001, first message deliveries weight W_{P_2} is 0.2, and the low message delivery rate penalty weight W_{P_3} is -0.1. It detects adversaries by their negative scores and prunes them from the honest network, as shown in Figure 4. The network can then work properly.

We then increase the time in local mesh W_{P_1} to 0.03 and the result is shown in the yellow line. In this case, the network is disrupted completely by the attack and no transactions can be broadcasted to all honest nodes in the network. This is because a high W_{P_1} makes sybil nodes earn high scores by just staying in the mesh, even though they don't process and propagate any messages. The network can not detect sybil nodes at all.

We also increase the low message delivery rate penalty weight W_{P_3} to -1 from the baseline, as shown in the green line. This case also prunes sybil nodes and shows the same performance as the baseline in the first 400 rounds. However, the high penalty makes some honest nodes also pruned due to low scores. As a result, the transactions being broadcasted to all honest nodes is decreased.

From the results above, we find that the weight of each component in the score function has to be tuned very carefully in order to defend against attacks efficiently.

4.2 How many adversarial nodes are needed for flash attack?

The next question is how many adversarial nodes are needed to launch flash attack successfully. By successfully, we mean it can decrease the throughput and latency greatly.

We then run a simulation on a 50-node network with 5 publishers, increasing the sybil ratio (=number of sybil nodes/number of total nodes) from 0 to 80%. The results show in Figure 6. The blue bar shows the number of honest components in the network and the red curve indicates the number of received transactions after 500 rounds. When the sybil ratio is low, the adversarial is hard to break down the honest network, so most of the generated transactions can be received by all honest nodes. If there are enough sybil nodes, they can partition the honest network into multiple components. As a result, the throughput is greatly reduced, only a small portion of transactions can be broadcasted to all honest nodes. Our simulation shows that $> 40\%$ being sybil nodes to implement a flash attack successfully with high probability.

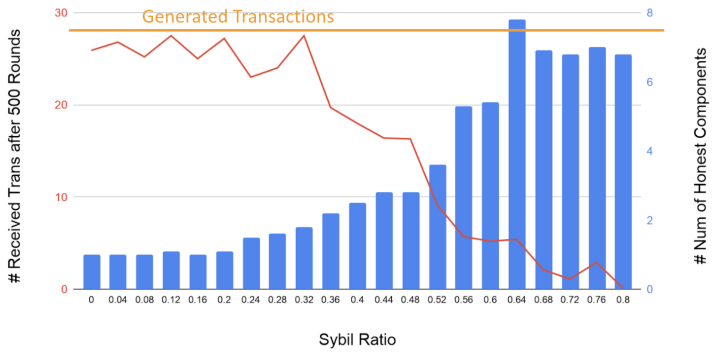

Figure 6: To launch a flash attack successfully with high probability, over 40% of nodes are needed to be adversaries. The blue bar shows the number of honest components in network and the red curve indicates the number of received transactions after 500 rounds.

5 Eclipse Attack

A eclipse attack is considered one of most harmful attacks because a successful attack allow an adversary to manipulate the view of a node regardless of any security guarantee at the upper layer. We tested the attack with two version of the codebase, one without Graft Sniping protection as described in the [1]. After we adapting the mitigation, we found it is nearly impossible to eclipse the network because the limitation of our simulation network that does not drop messages. To eclipse the targets in the new setup, we assume an adversary with

a strong computation resource, fast internet, ability to observe network and sometimes freeze a link in a network. In the simulation, we assume all honest node initially only connects to honest peers.

5.1 Graft Sniping Attack and Security Audit Mitigation

A Graft Sniping attack takes advantage of GossipSub1.0 that an honest node accepts all incoming messages even if the graft put the number of mesh connection over its limit. The excessive connection can only be pruned in the heartbeat round, which usually happens in 1 second. During this period, any sybil nodes can graft to the target and send fresh messages to the targeted node to boost score in order to survive in the next heartbeat. Meanwhile sybil nodes can connect to the honest peers of the target for wasting their bandwidth in attempts to slow down their messages deliver to the targeted nodes. Hence gradually the adversary can take over the entire mesh and eclipse the targets in the long run. The fix in the security audit is rather simple, every node can only accept the graft messages if it has spare mesh connection, otherwise all excessive graft messages are refused. With the new strategy in our simulation framework, all honest nodes will maintain their connections and build up their score. The adversary cannot use old methods to either waste the bandwidth or grafting.

5.2 Strong adversary with extra assumptions

To counteract the new protection scheme, we devise a more powerful adversary to accomplish two tasks: creating connection to target nodes, and breaking connection among target node and honest peers, which is depicted in Fig 7.

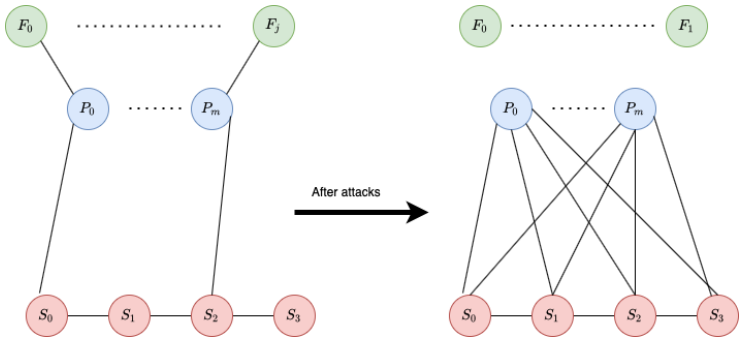

Figure 7: Eclipse strategy with powerful adversary. Blue nodes are publisher, green nodes are full nodes and red nodes are sybils

5.3 Creating connection to targets nodes

With the mitigation, although it is harder to graft many sybils nodes into the target mesh at once, there always one or two spots left empty. Sybil nodes will

takes some spots in some publishers and grow its score as long as targets are online. To survive to the next heartbeat, the sybils use their fast relay network to forward messages for multiple publishers they connect to. In the example in Fig 7, Sybil S_0 will gain high score in the publisher P_0 view by quickly delivering messages from P_m, and vice versa for S_2 with P_m. Hence there is a reliable way for sybil nodes to stick with targets. On the other hand, for completely eclipsing the targets, sybil nodes need to break connections among honest peers. Sybil nodes rely on the ability to freeze links among honest peers so the penalty term in the score function will eventually drive the score of a honest neighbor below zero, and which later will be substituted by other sybils nodes.

5.4 Attack scheduling

Section 5.3 gives a general method which adversary can use to eclipse the nodes, the execution needs extra cares on how to maximize the number of sybils connection around the targets and how to minimize the number of times to freeze a link between honest nodes.

For each targeted honest node, a new published message can be only used once to boost score for one sybil node, therefore an adversary need to be careful about how to distribute the chance evenly to all connected sybils, instead of building up scores for one sybil node. We conceptualize the chance to gain higher score in the form of tokens, every token corresponds to a targeted honest node, and any sybil node can request such token if it knows its score is close to 0. A token is returned whenever it reaches to some safety level.

To break connections among honest nods, adversary needs to decide when and how long it should freeze the links among honest peers. We associate the cost of an attack to be the number of freeze issued by the adversary to eclipse the targets. In the scheduling process, the adversary observes the scores for all mesh connections with the targets; if some scores become close to positive, the freezing attack is used until the scores associated with the honest peer drop well below 0.

5.5 Experiments

We setup a network of 100 nodes, with 2 publishers, 10 sybil nodes and 88 full nodes. The experiment runs in round which equals to 5 millisecond, and transaction is generated every 50ms per publisher. The heartbeat is set at 1 second, so that every honest node has to refresh its mesh every 200 rounds. Every honest nodes are randomly connected to 3 other honest nodes at the beginning, each honest node can maintain connections up to 7 peers simultaneously in one round.

In Fig 8, the top two plots show the number of eclipsed number and ratio of corrupted mesh connections for the two publishers, it is clear to see the eclipsing events only occur at the heartbeat round; in the second plot, we observe that the attack is most significant at the beginning, because at the beginning each targeted honest node has four spare mesh connections available for the

sybil nodes. The third and fourth plot show the number of freezing attack at a particular round in the entire network, the attacks are sporadic, which suggests adversary can continuously damage the system as long as the budget is enough. We further test the freeze cost for attacking 2, 3, 5 publishers and calculate average costs per target, they are 315, 592 and 1178 respectively. The increasing network activity among the publishers is one of reasons that explain the increasing cost of the attack, because that sybil nodes have to repeatedly freeze the link for removing the completely.

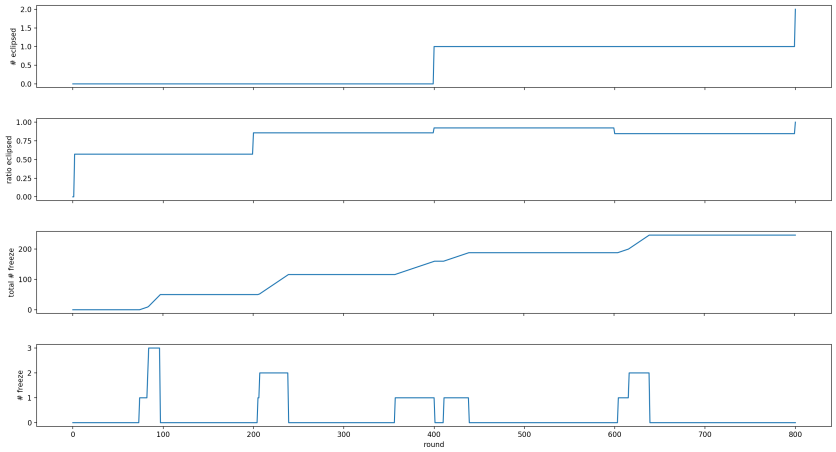

Figure 8: From top to bottom: number of eclipsed targets; ratio of corrupted edges and total number of mesh connections in the targets; cumulative freeze attack to attack both publishers; number of freeze attack to both targets at every round

6 Conclusion and Future Work

In this work, we created a testbed to understand the security of GossipSub subject to flash and eclipse attack in various setups. We note that using a linear combination of statistics to derive a score function will always give adversary means to prepare the attack. The weights in scores function are sensitive to performance and security, and it is difficult to develop a notion on what combination of parameters is the best. In addition, score functions are mainly designed for security, it will be interesting to see how score function can be used to improve throughput and latency. In the next step, we plan to develop a systematic way to design score functions that can optimize both security and performance.

References

[1] Protocol Labs. Gossipsub v1.1 protocol design + implementation: Security audit report. Available at `https://gateway.ipfs.io/ipfs/QmWR376YyuyLewZDzaTHXGZr7quL5LB13HRFnNdSJ3CyXu/Least%20Authority%20-%20Gossipsub%20v1.1%20Final%20Audit%20Report%20%28v2%29.pdf`.

[2] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and eth2.0 networks, 2020.