The Sybil Attack From the Attacker's Perspective

Andrew Rosen

November 22, 2014

Abstract

This paper explores the feasibility of performing naive Sybil attacks that completely occlude healthy nodes from each other. The vulnerability of Distributed Hash Tables to Sybil attacks and Eclipse attacks has been well known for some time. However, these vulnerabilities has often been explored in a theoretical sense, assuming the attacker is a global adversary from the beginning, nigh-omniscient and omnipotent. This paper examines, from a analytical and practical perspective, how valid that assumption is.

We examine the amount of computational effort required to become a global adversary starting as a non-global adversary. We do this by analyzing the amount of time it takes an attacker with a given IP to choose a port to obtain a desired hashkey, a process we call *potatoing*. We present potatoing to emphasize the ease of this attack, but also demonstrate potential non-security uses of potatoing that are beneficial to DHT load-balancing.

1 Introduction

One of the key properties of structured peer-to-peer (P2P) systems is the lack of a centralized coordinator. P2P systems remove the vulnerability of a single point of failure and and the susceptibility to a denial of service attack (cite original Sybil paper), but in doing so, open themselves up to new attacks. Completely decentralized P2P systems are vulnerable to *Eclipse attacks*, whereby an attacker completely occlude healthy nodes from one another. This prevents them from communicating without being intercepted by the attacker.

One way to accomplish this attack is to perform a *Sybil Attack* [4]. In a Sybil attack, the attacker creates multiple malicious virtual nodes in order to disrupt the network. If enough malicious nodes are injected into the system, the majority of the nodes will be occluded from one another, successfully performing an Eclipse attack.

Security analyses typically assumes an adversary using a Sybil is omniscient and can inject virtual nodes wherever he chooses in a reasonable amount of time. Our goal is to demonstrate how veracity of that assumption by doing analysis and simulations.

Sybil attacks represent a significant threat to the security of any distributed system. Many of the analyses [1] on Tor [3] emphasize the vulnerability of Tor to the Sybil attack. This threatens the anonymity of Tor users, particularly those living in countries with peculiar notions about personal privacy CITATION: WHO USES TOR.

P2P systems like BitTorrent are essential to a wide variety of users. BitTorrent, for instance, is the *de facto* platform for distributing large files scalably among tens of thousands of clients An estimated 20 million peers use BitTorrent daily for sharing and retrieving Linux and BSD-based distros, offline copies of Wikipedia, and personal backup Current research demonstrates BitTorrent is vulnerable to the Sybil attack and a persistent attack disabling BitTorrent would e highly detrimental to many users, but especially developers and system administrators [1].

There have been many suggestions on how to defend against Sybil attack, but there is no agreed upon "silver bullet" among researchers that should be implemented for every distributed application [5] Part of this is surely influenced by the only surefire way to defend against a Sybil attack is to introduce a trusted authority to certify and/or bind identities. This solution potentially removes the Sybil attack, but reintroduces vulnerabilities to denial of service attacks, bringing us full circle. Another reason is that there is no single solution for all platforms is that not every solution is compatible with each distributed platform.

Despite the threat represented by the Sybil attack and the research done on the subject, little research has been done from the perspective of an adversary. We sought to rectify this, both to reemphasize the threat of the Sybil attack, but also because this examination introduces some interesting graph theory problems.

Our work presents the following contributions:

- We first discuss the mechanics of performing a Sybil attack and analyze the theoretical effort needed to bring to bear to perform the Sybil attack.
- We present our simulations which show how quickly even a naive and inefficient Sybil attack can compromise a system. We also discuss how a

[1] Perhaps the only reason that BitTorrent hasn't be attacked in such a way as to render it unusable is that those capable of doing so rely heavily upon it. A sobering thought.

more intelligent attack can be geared to each of the more popular DHT topologies.

- We analyze an interesting graph coloring problem that an attacker needs to solve if the attack is to remain undetected.
- We discuss the implications of our work, specifically how the techniques we developed to perform the attack can be used for automatic load balancing

2 Formal Analysis

[2]

The birthday attack analysis says given so many elements, likelyhood of collision between any of these elements. The potato attack says given this region, what is likelyhood i can find something in this region. It's a different analysis since I'm looking for 1 attacker colliding with one specified region at a time.

Suppose we have a DHT with N members in it, with the hashspace of $[0, 2^{160})$. The case of small N is ignored, since they are trivial even when unbalanced. We can assume that, for a large enough N, node IDs will beclose to evenly distributed accross the network, meaning there will be $\approx \frac{2^{160}}{N}$ hashkeys between each node ID.

Size of bin is

$$bin = \frac{H}{N}$$

The probability P of an attacker finding an hashkey that lands in the range with the range (n, m) is

$$P \approx \frac{n-m}{H} \cdot num\_ips \cdot num\_ports$$

making it equivalent to

$$P \approx \frac{1}{N} \cdot num\_ips \cdot num\_ports$$

The chances of compromising an entire network of N nodes that partition the entire network.

$$c \approx 1 - (1 - \frac{1}{N})^{P*I})$$

Alternatively, we can view this as doing a birthday attack in progress with different probabilities. EG, we've generated $\frac{h}{N}$ values already, how many more do we need?

3 Simulations

Simulations were performed on a computer with consumer-grade budget hardware.

3.1 Experiment 1:Potatoing 2 random nodes

Our initial experiment was designed to establish the feasability of injecting in between two random nodes.

Each trial, we generated two victims with random IP/port combinations, and an attacker with a random IP. The experiment was for the attacker to find a hashkey in between the two victim's key, from the lowest to highest.

The amount of time to potato two random hashkeys was 29.6218323708 microseconds, and was achevied 99.996% of the time.

3.2 Experiment 2: Nearest Neighbor Eclipse via Sybil

The objective of the second experiment is to completely ensnare a network using a Sybil attack, starting with single node.

3.3 Experiment 3: Fully Complete Eclipse via Sybil

The previous experiments of

In some of the below frameworks, it is more efficient to perform an Eclipse attack by falsely advertising, rather than injecting

Any application built using a DHT must be address its vulnerabilities to the Eclipse and Sybil attacks

Security is not something that is thought about for a DHT, unless the DHT is specifically made to be secure against X. Or it's left to the applications

3.3.1 Chord

Most of the above attacks

Chances the finger is already covered:

3.3.2 Kademlia

3.3.3 Plaxton Based networks

More efficient to lie.

4 Masking the Attack

Now that we have established that a Sybil attack can be performed with great ease, our focus now turns to avoiding detection.

We need a different IP for each point surrounding our victim. In the Nearest-Neighbor attack, we need a

We can reduce this into an interesting graph coloring problem.

The hard maximum, in general, is m seperate IP addresses, one for each bit in a $\log n$ routing/routing table DHT. Recall that the vast

5 Simple Load Balancing Injection Framework

Costs: Need to hold 15000ish hashkeys, effectily 160-bit numbers

References

[1] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM, 2007.

[2] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In *Advances in Cryptology-Eurocrypt 2004*, pages 401–418. Springer, 2004.

[3] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[4] John R Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[5] Brian Neil Levine, Clay Shields, and N Boris Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA*, 2006.